

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

AMPL: AN EXCLUSIVE TECHNOLOGY
FOR DecisionNet

by

Michael P. Casey

March, 1999

Thesis Advisor: Hemant K. Bhargava
Second Reader: Gordon Bradley

Approved for public release; distribution is unlimited.

19990512 025

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 1999	3. REPORT TYPE AND DATES COVERED Master's Thesis		
4. TITLE AND SUBTITLE AMPL: AN EXCLUSIVE TECHNOLOGY FOR DecisionNet		5. FUNDING NUMBERS		
6. AUTHOR(S) Casey, Michael P.				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER		
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (maximum 200 words) This thesis describes the design and implementation of an exclusive technology agent for DecisionNet -- a distributed decision support technology marketplace for the World Wide Web. Unlike the independent technologies presently supported by DecisionNet, exclusive technologies require software agents to facilitate interactions between their providers and consumers. The exclusive technology agent described herein would allow use of the AMPL mathematical modeling language through the DecisionNet environment and provide database support for AMPL objects. An AMPL agent for DecisionNet provides a searchable repository to facilitate sharing and reuse of AMPL objects. It also provides a means for clients to formulate and execute AMPL problems without having to purchase AMPL software or learn the AMPL language. The data structure described in this thesis has three distinct types of exclusive objects: model schema, dataset, and solver. A complete AMPL problem contains one object of each type. An AMPL agent assists DecisionNet technology providers in the registration, modification, and withdrawal of AMPL objects, and assists DecisionNet consumers in the selection and execution of AMPL problems.				
14. SUBJECT TERMS DecisionNet, Decision Support, World Wide Web, Common Gateway Interface, AMPL, Software Agent, Exclusive Technology			15. NUMBER OF PAGES 140	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18 298-102

Approved for public release; distribution is unlimited.

**AMPL: AN
EXCLUSIVE TECHNOLOGY
FOR DecisionNet**

Michael P. Casey
Lieutenant, United States Navy
B.S., University of Rochester, 1989

Submitted in partial fulfillment
of the requirements for the degree of

**MASTER OF SCIENCE IN
INFORMATION TECHNOLOGY MANAGEMENT**

from the

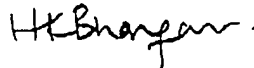
**NAVAL POSTGRADUATE SCHOOL
March 1999**

Author:

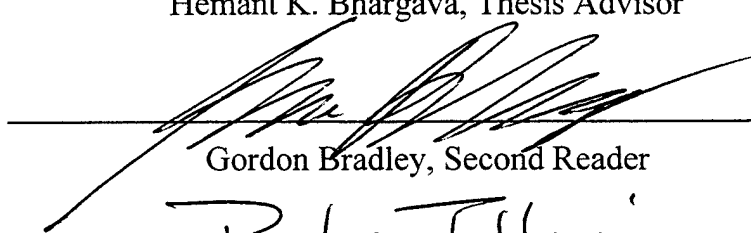


Michael P. Casey

Approved by:



Hemant K. Bhargava, Thesis Advisor



Gordon Bradley, Second Reader



Reuben T. Harris, Chairman
Department of Systems Management

ABSTRACT

This thesis describes the design and implementation of an exclusive technology agent for DecisionNet -- a distributed decision support technology marketplace for the World Wide Web. Unlike the independent technologies presently supported by DecisionNet, exclusive technologies require software agents to facilitate interactions between their providers and consumers. The exclusive technology agent described herein would allow use of the AMPL mathematical modeling language through the DecisionNet environment and provide database support for AMPL objects.

An AMPL agent for DecisionNet provides a searchable repository to facilitate sharing and reuse of AMPL objects. It also provides a means for clients to formulate and execute AMPL problems without having to purchase AMPL software or learn the AMPL language.

The data structure described in this thesis has three distinct types of exclusive objects: model schema, dataset, and solver. A complete AMPL problem contains one object of each type. An AMPL agent assists DecisionNet technology providers in the registration, modification, and withdrawal of AMPL objects, and assists DecisionNet consumers in the selection and execution of AMPL problems.

TABLE OF CONTENTS

I. INTRODUCTION	1
A. THE DecisionNet PROJECT	1
B. EXCLUSIVE TECHNOLOGIES FOR DecisionNet	1
1. Registration, Modification, and Withdrawal of Exclusive Objects ..	2
2. Execution of Exclusive Technology Objects	2
C. THE AMPL LANGUAGE AND ENVIRONMENT	3
II. FUNCTIONAL SPECIFICATIONS	5
A. GENERAL SPECIFICATIONS	5
B. USER INTERFACE	6
C. DATABASE	7
1. Model Schemas	7
2. Datasets	7
3. Solvers	8
D. INTEGRATION WITH DecisionNet	8
1. User Identification and Authentication	8
2. Object Cataloging	9
3. A Seamless User Interface	9

III. DESIGN SPECIFICATIONS	11
A. DATABASE DESIGN.....	11
B. AMPL OBJECT REGISTRATION PROCESS	11
C. AMPL OBJECT EXECUTION PROCESS	13
D. AMPL OBJECT UPDATE PROCESS	17
E. AMPL OBJECT WITHDRAWAL PROCESS	18
IV. IMPLEMENTATION NOTES.....	21
A. AMPL DATABASE	21
B. AMPL PROVIDER SCRIPTS.....	22
1. AMPLREG	22
2. ASOLVREG	22
3. ASLVREG2	23
4. ADSREG.....	23
5. ADSREG2.....	23
6. AMODREG	23
7. AMODREG2	24
8. UPDATE1	24
9. UPDATE2.....	24
10. WITDRAW1	25
C. AMPL CONSUMER SCRIPTS	25
1. AMPLEXEC.....	25

2.	AMPLEXE2.....	26
3.	AMPLEXE3.....	26
4.	AMPLEXE4.....	26
5.	AMPLEXE5.....	26
6.	AMPLEXE6.....	27
7.	AMPLEXE7.....	27
8.	AMPLEXE9.....	27
D.	AMPL SOLVER AGENT SCRIPT	28
V.	CONCLUSIONS AND RECOMMENDATIONS	31
A.	AMPL AS AN EXCLUSIVE TECHNOLOGY FOR DecisionNet.....	31
B.	SUGGESTIONS FOR FURTHER DEVELOPMENT.....	32
	APPENDIX A. DATA DICTIONARY	35
	APPENDIX B. SOURCE CODE	37
	LIST OF REFERENCES.....	125
	INITIAL DISTRIBUTION LIST.....	127

ACKNOWLEDGMENT

The author would like to acknowledge the financial support of the U.S. Army Artificial Intelligence Center under Grant MIPR6GNGS00087.

The author wants to thank Professors Bhargava and Bradley for their expertise and guidance in performing this research. Steven Earley, Vera Parker, and Clay Tettelbaugh were a pleasure to work with. Mark Murphy provided invaluable assistance in the writing process. The author extends sincere thanks to each of them.

I. INTRODUCTION

This thesis describes the design and implementation of an exclusive technology agent for DecisionNet, [1,2] a distributed decision support technology marketplace for the World Wide Web. This exclusive technology agent would allow use of the AMPL mathematical modeling language [3] through the DecisionNet environment and provide database support for AMPL objects. This project could also serve as a framework for incorporating additional exclusive technologies into DecisionNet.

A. THE DecisionNet PROJECT

DecisionNet is a WWW-based marketplace that brings together providers and consumers of decision technologies and resources. Its purpose is to reduce the informational, logistical, and financial barriers to interaction by the providers and consumers it serves.[1]

DecisionNet acts as a searchable registry of decision support technologies available from a variety of providers to decision technology consumers at large. In order for consumers and providers of decision support technology to transact, they must first be brought into contact. DecisionNet helps bring them together by maintaining an easily accessible and searchable registry of electronic decision support products and services.[1]

DecisionNet facilitates an alternative to the traditional logistics of software distribution by allowing consumers of decision support technologies to access those technologies on the providers' computational platforms via the WWW. By eliminating the need to sell software licenses and physically distribute software media, this arrangement can save time and money for both providers and consumers. It also gives consumers a means to test and evaluate candidate technologies with a minimum of time, cost, and commitment.[1]

B. EXCLUSIVE TECHNOLOGIES FOR DecisionNet

DecisionNet technologies fall into two categories, independent and exclusive. Independent technologies are those which require no further assistance from DecisionNet

once the provider and consumer are brought into contact with one another. Exclusive technologies are those which require the use of an agent throughout the consumer-provider interaction.[1]

The fundamental building block of an exclusive technology is an exclusive object. An exclusive object acting alone does not provide a useful function to the consumer. Depending on the structure of the exclusive technology, two or more objects must interact with each other to create an instance of the technology. The term *instance* refers to a temporary combination of complementary exclusive objects capable of generating usable information for the consumer.

The example of linear programming as an exclusive technology illustrates the relationships between exclusive objects. A *model schema* object contains the objective function, constraint equations, and variables of a particular type of linear program. A *dataset* object contains a set of specific values to be inserted into the model schema to instantiate a specific, complete linear program. A *solver* object processes the complete linear program, yielding its solution and other related information.

An exclusive technology agent performs several related functions.

1. Registration, Modification, and Withdrawal of Exclusive Objects

The DecisionNet agent performs the basic functions of object registration, modification, and withdrawal of independent technologies.[1] It also performs these functions for exclusive technologies, but must do so in conjunction with the appropriate exclusive technology agent. The latter applies rules of object interaction to these functions. For instance, one dataset object may have been designed for a specific model schema, yet may also be compatible with certain other model schemas. The exclusive technology agent should retain such information and update it when objects are modified or withdrawn.

2. Execution of Exclusive Technology Objects

The DecisionNet agent handles an execution request by sending it to the appropriate host server address, along with all required information concerning the request and the consumer who made it.[1]

C. THE AMPL LANGUAGE AND ENVIRONMENT

The AMPL mathematical modeling language provides a means of formulating optimization problems in a notation that facilitates solution using readily available solver algorithms. The AMPL syntax is concise, yet easily understood by the reader familiar with algebraic notation and the English language. [3]

The AMPL mathematical modeling environment consists of one or more solver algorithms, a command interpreter, and a set of algorithms for processing model instances and solution results. The command interpreter allows the user to provide model instances, select and configure solver algorithms, invoke solvers, and format the solution output. [3]

AMPL has traditionally been implemented with a command line interface on MS DOS and various UNIX platforms, although versions for Microsoft Windows are now available as well. The command line interface provides two modes of operation. In interactive mode, the user can enter individual AMPL commands or provide text files containing AMPL commands, which are processed by the AMPL interpreter as if typed individually by the user. In batch mode, the user invokes AMPL from the command line, along with the name of a batch file containing a complete set of AMPL commands to carry out the user's intended actions. [3]

A version of AMPL for Microsoft Windows provides a friendlier interface for the user, and transparently calls the MS-DOS command-line version of AMPL to process the user's commands.

II. FUNCTIONAL SPECIFICATIONS

Functional specifications fully describe the system's requirements in the most general manner possible. [4] This allows maximum flexibility later, in the system design phase. The functional specifications described below are the product of the system analysis phase of the AMPL agent project. They combine the various restraints that would be placed on the AMPL agent by various sources, including: the nature of the AMPL language, the AMPL solvers that would be used in the project, and the interface specifications for DecisionNet. The functional specifications also incorporate certain goals of the project, such as shielding AMPL agent users from unnecessary technical details.

A. GENERAL SPECIFICATIONS

An AMPL agent for DecisionNet must support three types of AMPL objects: *model schemas*, *datasets*, and *solvers*.

A model schema defines the general form of an AMPL problem. It contains a series of statements describing the following: one or more variables, an objective function, and the structure of the data that must be provided to generate a solution. It may additionally describe one or more constraints that the variable(s) must satisfy. It consists of text information only, and is subject to the syntax requirements of the AMPL language. [3]

A dataset contains actual values for variables and constants in a model schema, formatted in accordance with the structure specified in the model schema. Like a model schema, it too consists only of text information, and follows AMPL syntax. When a dataset is combined with an associated model schema, they describe the specific form of an AMPL problem. [3]

A solver contains one or more computational algorithms capable of generating a solution to the specific form of an AMPL problem (model schema plus dataset). It also has the ability to configure and choose among its solver algorithms, as well as generate

solution output in text form as directed by the user. [3]

In traditional implementations of AMPL, a program may contain interspersed data statements, model statements, and solver commands, as long as they appear in a logical order. Additionally, the data for a particular problem can be broken up among multiple datasets. [3]

In a DecisionNet AMPL implementation, the model schema, dataset, and solver must be distinct objects. A complete DecisionNet AMPL problem consists of exactly one model schema, one dataset, and one solver. A model schema has a fixed data structure, so only datasets using that same structure will be compatible. However, multiple model schemas may use the same data structure, enabling a given dataset to be used with a number of different model schemas.

The operations that may be performed on AMPL objects should be the same as those performed on independent DecisionNet objects. Namely, a provider should be able to register AMPL objects, and to update or withdraw any object registered to that provider. [1] A consumer should be able to execute any registered AMPL object. To do this, the consumer must define a complete AMPL problem by selecting a model schema, dataset, and solver in any logical order. The consumer must use a registered solver, but may provide an unregistered model schema and/or dataset for use with it on a one-time basis.

B. USER INTERFACE

The AMPL agent should communicate with users via hypertext markup language (HTML) pages and forms. Thus, to access the AMPL agent, a user would need an Internet connection and WWW browser software. These requirements apply to all DecisionNet users, [1] so anyone capable of accessing DecisionNet could also access the AMPL agent. However, significant additional requirements would be necessary for providers of AMPL solvers. Because an AMPL solver would be maintained by its provider, vice being stored in the AMPL database as will be discussed later, the solver provider would also have to maintain a WWW server.

C. DATABASE

An AMPL agent would require a relational database to store information on registered AMPL objects. [6] All AMPL object attributes, except those common to all DecisionNet objects, should be stored in this database. The majority of these attributes are *metainformation*. This term refers to information that is unrelated to the actual content of an object, but determines how the DecisionNet and AMPL agents handle the object. [5] The specific additional metainformation (besides that stored in the DecisionNet database) required for the three types of AMPL objects are described below.

1. Model Schemas

For a model schema, additional metainformation includes a pointer to a default solver. The default solver is included mainly to help the consumer choose a solver when indifferent or unfamiliar with AMPL. Some model schemas may require special solver algorithms, such as nonlinear programming or integer programming. [3] In such cases, the provider of the model schema should specify a default solver that is capable of performing that particular type of computation.

In addition to metainformation, the database should also store the actual text of the model schema, as well as an appropriate output script. An *output script* is a sequence of AMPL commands that specify and configure an AMPL solver algorithm, invoke the solver, and generate formatted solution output as specified by the provider of the model schema. An output script is necessary for several reasons. First, AMPL does not provide adequate solution output unless specific additional commands are provided. Second, some model schemas may need to temporarily reconfigure the solver, or require the use of a particular solver algorithm which is not normally used. [3] Third, AMPL consumers on DecisionNet should be shielded from such low-level decisions to reduce the level of AMPL knowledge required of them.

2. Datasets

The stored metainformation for a dataset includes a pointer to a default model schema. This should normally be the model schema for which the dataset was designed.

Since an AMPL database may hold several related model schemas that are compatible with a given dataset, a consumer who chooses to use a registered dataset must be afforded the opportunity to choose a model schema to accompany it. In such cases, the default model schema should be suggested to the consumer, to reduce the level of AMPL knowledge required.

In addition to metainformation, the actual text of the dataset is stored in the AMPL database. As with model schemas, this prevents the dataset provider from having to provide a separate server.

3. Solvers

The stored metainformation for a DecisionNet AMPL solver would consist solely of the Internet Uniform Resource Locator (URL) to be used to access the solver. For example, *http://Sample.com/DecisionNet/AMPL/Server.exe* specifies the access protocol, server, path, and filename of a fictitious DecisionNet AMPL solver.

D. INTEGRATION WITH DecisionNet

As an exclusive DecisionNet language, such an implementation of AMPL should rely on DecisionNet for all support functions which are not unique to AMPL. Doing so would reduce the burden on providers of exclusive DecisionNet technologies, and maximize continuity across the interface between DecisionNet and its exclusive technologies.

1. User Identification and Authentication

DecisionNet handles the registration, login, and logout of consumers and providers. This helps prevent unauthorized access to the system, and provides a unique 15-character identifier for each user. [1] Whenever a user chooses to access an AMPL object from within DecisionNet, DecisionNet would verify the user's active status, then pass the user's identifier to the AMPL agent along with the particulars of the request. Because all access to the AMPL agent would be via DecisionNet, the AMPL agent need not perform a redundant check of a user's status.

Since DecisionNet assigns and controls users' identifiers, an AMPL agent should

also use these same identifiers to uniquely identify the provider of each AMPL object registered with DecisionNet.

2. Object Cataloging

Every object registered with the AMPL agent would also be registered with DecisionNet. The DecisionNet object registry would manage the object identifiers for all exclusive and independent technologies registered. (An object identifier consists of the 15-character identifier of its provider, along with a 3-character serial code.) [1] The DecisionNet object registry would also store, for each registered AMPL object, all object attributes that are not particular to AMPL. Examples of such attributes include: technology name, object type, functional area, and problem area. [1] Besides avoiding duplication of stored data between the DecisionNet and AMPL agent databases, this arrangement would allow registered AMPL objects to be searched as an integral part of DecisionNet's "Yellow Pages" of decision support technologies.

3. A Seamless User Interface

The WWW interface of DecisionNet would provide a "front end" for the AMPL agent's user interface, as well as a template for the look and feel of the AMPL agent's user interface. Therefore, a user must access DecisionNet in order to reach an AMPL agent, but the transition from DecisionNet to the AMPL agent and back again should be transparent to the user.

III. DESIGN SPECIFICATIONS

The design specifications are the result of the system design phase which divides the system into a logical structure of related functions, or modules. [4] The system is naturally divided into four major processes: registration, execution, update, and withdrawal. These processes are further broken down into modules, such that each module performs a specific function in support of its parent process.

A. DATABASE DESIGN

The database should consist principally of one table for each type of AMPL object. Each table should hold data for each registered AMPL object of the appropriate type. Each table should contain, at a minimum, the appropriate metainformation as described above in the functional requirements. The table for model schemas should contain the text of the model itself, as well as an output script. The table for datasets should contain the text of the dataset itself.

Each table should use DecisionNet's unique technology identifier (the 15-character ProviderID plus the three-character TechID) as its primary key. This facilitates data exchange between the DecisionNet and AMPL agents and their databases.

B. AMPL OBJECT REGISTRATION PROCESS

The registration process for DecisionNet AMPL objects is depicted in Figure 1. The DecisionNet agent would handle the first steps of registration by accepting from the provider all metainformation that is not specific to AMPL. Included in this metainformation is the type of object to be registered (i.e., model schema, dataset, solver, etc.) and the technology type (i.e., independent, AMPL, or another exclusive type). If the technology to be registered is independent, DecisionNet's *REGTECA* module merely stores this metainformation in its database. If the technology is an AMPL object or another exclusive object, *REGTECA* looks up the appropriate URL for registering that type of object, then sends the provider to that URL. The technology ID, which contains

the provider's identifier and a serial number, is passed to the exclusive agent via hidden form fields. The metainformation collected up to this point is stored by *REGTECA* in a temporary database table until the object is fully registered. This prevents consumers from accessing the object during the remainder of the registration process. [1]

The first module in the AMPL agent's registration process, *AMPLREG*, verifies from DecisionNet's temporary technology table that the object to be registered is a valid AMPL object type. *AMPLREG* then passes the technology ID to the appropriate module, according to object type: *AMODREG* for a model schema, *ADSREG* for a dataset, or *ASOLVREG* for a solver.

1. Model Schemas

The *AMODREG* module provides an HTML form for the user to submit the model text and output script text, and to select a default solver from a list of all registered AMPL solvers. The *ASLVREG2* module receives this information from the user and stores it in the model table of the AMPL database. Then it passes the technology ID to DecisionNet's *TECHDONE* module, which transfers the metainformation collected by *REGTECA* from the temporary storage table to the regular table for registered objects. The registration process is then complete.

2. Datasets

The *ADSREG* module provides an HTML form for the user to submit the dataset text and to select a default model schema from a list of all registered AMPL model schemas. The *ADSREG2* module receives this information from the user and stores it in the dataset table of the AMPL database. Then it passes the technology ID to DecisionNet's *TECHDONE* module, which completes the registration process as described above.

3. Solvers

The *ASOLVREG* module informs the provider of the various requirements for registered DecisionNet AMPL solvers, then provides an HTML form for the user to submit the URL for the solver to be registered. The provider may opt to stop the registration process at this point and return to DecisionNet's *MENU* module. Otherwise,

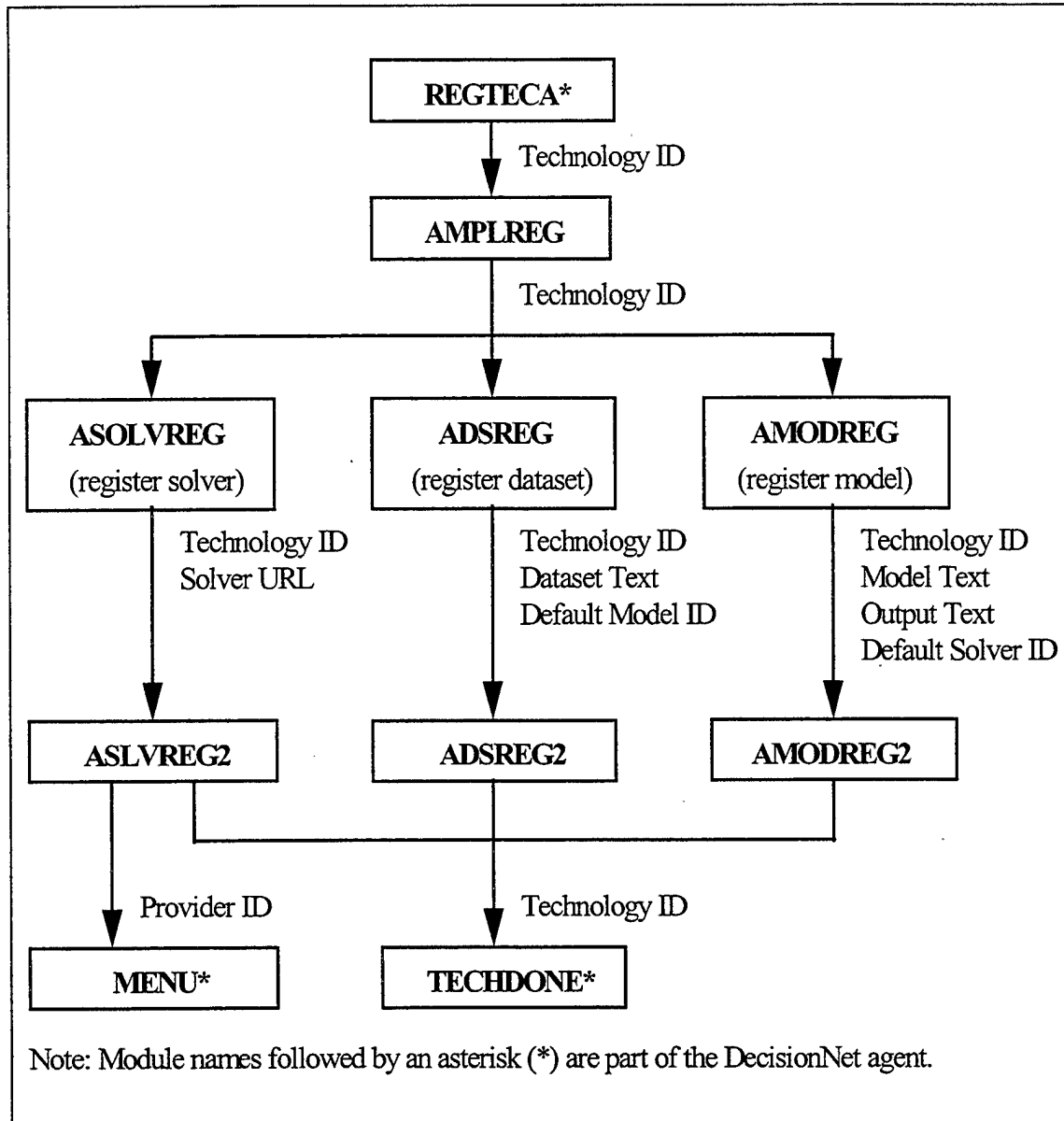


Figure 1. AMPL Object Registration Process

the *ASLVREG2* module receives the solver URL from the user and stores it in the solver table of the AMPL database. Then it passes the technology ID to DecisionNet's *TECHDONE* module, which completes the registration process as described above.

C. AMPL OBJECT EXECUTION PROCESS

The execution process for DecisionNet AMPL objects is depicted in Figure 2. The

DecisionNet agent handles the first step of execution by accepting from the consumer the technology ID to be executed. DecisionNet's EXECIND module looks up the appropriate URL for executing that type of object (e.g., AMPL), then sends the consumer to that URL. The technology ID, which contains the object provider's identifier and a serial number, is passed to the AMPL agent via hidden form fields. [1]

The first module in the AMPL agent's registration process, *AMPLEXEC*, verifies from DecisionNet's technology table that the object to be executed is a valid AMPL object type. *AMPLEXEC* then passes the technology ID to the appropriate module, according to object type: *AMPLEXE2* for a model schema, *AMPLEXE3* for a dataset, or *AMPLEXE4* for a solver.

1. Model Schemas

The *AMPLEXE2* module generates an HTML form that allows the consumer to either select a registered AMPL dataset from a list, or to submit a custom dataset directly. This module also looks up the URL of the default solver for this model schema, and passes it along to the *AMPLEXE5* module with the consumer's dataset input. If the consumer provided a custom dataset, *AMPLEXE5* stores it in a database table reserved for temporary datasets. It then passes to the *AMPLEXE9* module the identifiers for the model schema and dataset, along with the solver URL.

2. Datasets

The *AMPLEXE3* module generates an HTML form that allows the consumer to either select a registered AMPL model schema from a list, or to submit a custom model schema directly. This module also looks up the URL of the default solver for this model schema, and passes it along to the *AMPLEXE6* module with the consumer's dataset input. If the consumer provided a custom dataset, *AMPLEXE6* stores it in a database table reserved for temporary datasets. It then passes to the *AMPLEXE9* module the identifiers for the model schema and dataset, along with the solver URL.

3. Solvers

The *AMPLEXE4* module generates an HTML form that allows the consumer to either select a registered model schema, select a registered dataset, or submit a custom

model schema and a custom dataset. This module also looks up the URL of the selected solver, and passes it along to the *AMPLEXE7* module with the consumer's model schema and dataset selections. *AMPLEXE7* then performs one of three actions, depending on the input it receives. If the consumer specified a model schema to be used with the selected solver, *AMPLEXE7* passes the solver's URL and the model schema's identification code to *AMPLEXE2*. If the consumer specified a dataset be used with the selected solver, *AMPLEXE7* passes the solver's URL and the dataset's identification code to *AMPLEXE3*. If the consumer opted to submit a custom model schema and dataset, *AMPLEXE7* stores them in the *TMODEL* and *TDATASET* tables, respectively, then passes their identification codes along with the solver's URL to *AMPLEXE9*.

The *AMPLEXE2* and *AMPLEXE3* modules behave differently when called by *AMPLEXE7* vice *AMPLEXEC*, in that they use the URL of the user-selected solver in place of the URL of the default solver for the chosen model schema. The presence of a solver URL in the input to these modules triggers the change in behavior. Similarly, these modules pass the solver URL directly to their successors, which alter their behavior in a similar fashion.

When the execution process reaches module *AMPLEXE9*, the consumer has selected or provided a model schema and a dataset, and may have also specified which registered solver is to be used. The *AMPLEXE9* module displays the model schema text and output script, as well as the dataset text, giving the consumer an opportunity to examine and modify them before executing the complete AMPL problem. *AMPLEXE9* sends this final version of the model schema and dataset to the AMPL solver agent selected in the previous modules. *AMPLEXE9* also sends the solver agent a sequential serial number to uniquely identify the solver agent output later.

The AMPL solver agent receives the model schema and dataset from *AMPLEXE9*, then combines them into an AMPL batch file and executes it. The solver agent captures the AMPL output into a text file, names that file with the serial number it receives from *AMPLEXE9*, then sends this file back to the AMPL agent's server. The solver then sends the consumer a HTTP link to the output file's URL on the AMPL

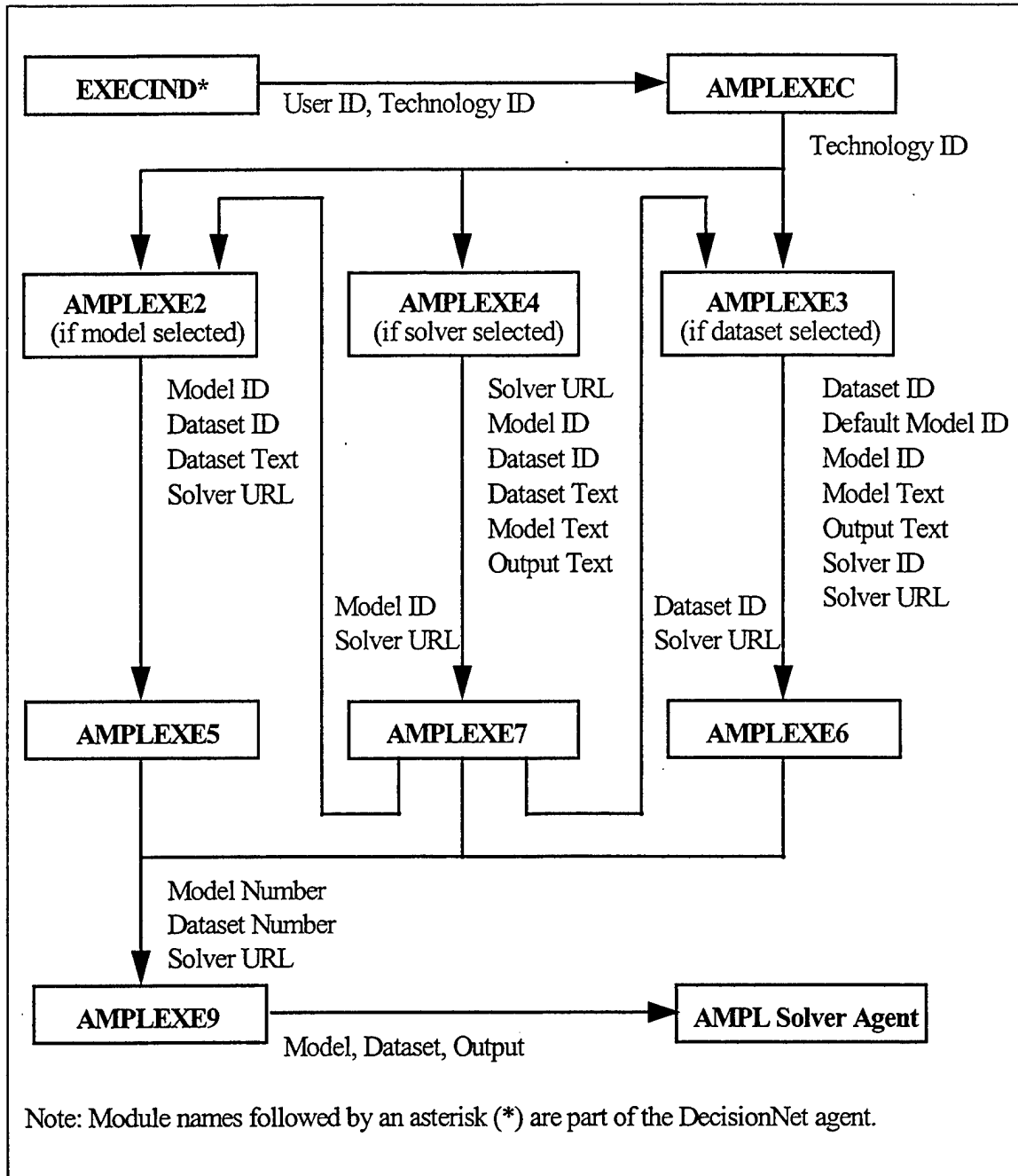


Figure 2. AMPL Object Execution Process

agent's server. This arrangement is more complicated than sending the output file from the AMPL solver agent directly to the consumer, but it also has several distinct advantages. First, it gives DecisionNet the capability to monitor how many times each AMPL solver agent is used and by whom. This capability is not currently required, but

may be in the future, especially if consumers are to be charged for services received. Second, it allows AMPL output files to be archived for a designated time period, without placing this burden on the provider of the solver. Third, it sends all AMPL output files to one central location with a common serial numbering system. This prevents any confusion which might otherwise occur when using multiple AMPL solver agents in a single DecisionNet consumer session.

D. AMPL OBJECT UPDATE PROCESS

The update process for DecisionNet AMPL objects is depicted in Figure 3. Updating a DecisionNet AMPL object consists of two major subprocesses. First, the object's provider uses the DecisionNet agent to update all data that is not specific to AMPL. However, certain attributes such as the object's unique identifier and the object type (independent, AMPL, etc.) may not be altered, as doing so would essentially create a new object rather than update an existing one. This subprocess is the same one used to update independent DecisionNet objects. When this subprocess is complete, the DecisionNet agent's *UPDTTECB* module checks to see whether the object being updated is independent or exclusive. If the object is independent, the update process is complete. If the object is an AMPL object, *UPDTTECB* sends the object's unique identifier to the AMPL agent's module *UPDATE1*, which begins the next subprocess.

The second subprocess updates all attributes that are specific to AMPL. These include all of the attributes collected by the AMPL agent when the object was first registered. *UPDATE1* checks the DecisionNet database's *TECHNOLO.DB* table to determine which type of object (model schema, dataset, or solver) is being updated. It then checks the corresponding table in the AMPL database to make sure the object's attributes are stored there. If no corresponding entry is found in the AMPL database, then the object's provider changed the object from one type to another in the previous step. Since the three types of AMPL objects differ greatly in structure and function, it is not appropriate to change the type of an AMPL object via the update process. Therefore, the *UPDATE1* module will display a warning message that explains the situation and

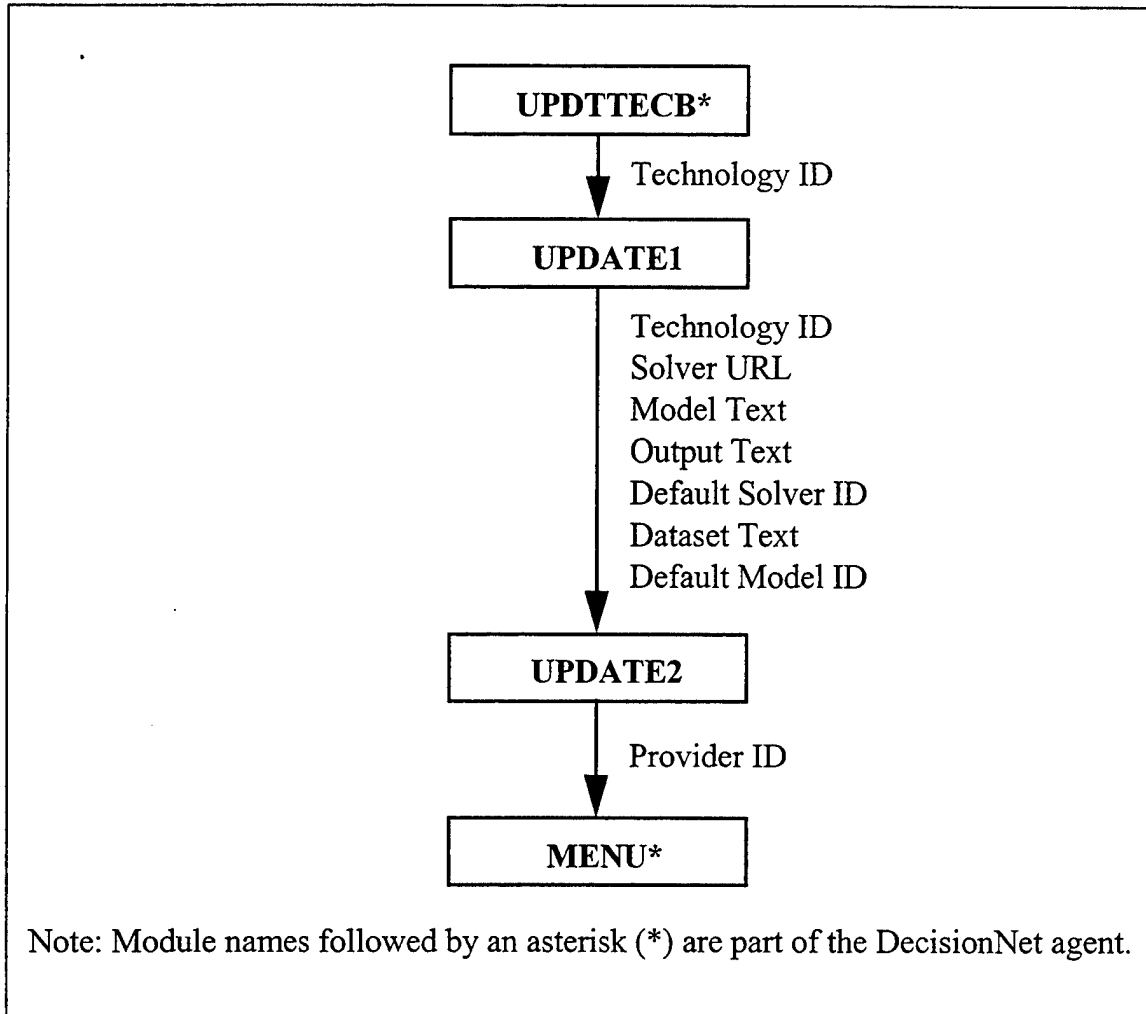


Figure 3. AMPL Object Update Process

instructs the provider to restore the object to its original type.

If the object's record is found where expected in the AMPL database, the *UPDATE1* module displays all AMPL-related attributes for that object for examination and modification by the provider. The updated attributes are received by the *UPDATE2* module, and stored in the AMPL database. At this point, the update process is complete, and *UPDATE2* then sends the provider back to the DecisionNet module *MENU*.

E. AMPL OBJECT WITHDRAWAL PROCESS

The withdrawal process for DecisionNet AMPL objects is depicted in Figure 4.

Withdrawing an object from the DecisionNet and AMPL registries is more complex than the updating an object. When withdrawing an AMPL object, the provider must obey certain "business rules."

- That solver which has been designated as the default solver for the DecisionNet AMPL system may not be withdrawn until this designation is transferred to another registered solver. If no default solver were available for the system, then every consumer would be forced to make a conscious choice among the remaining registered solvers. More importantly, preserving the default solver prevents withdrawal of **all** solvers, which would render the DecisionNet AMPL system unable to process AMPL problems.
- If a registered AMPL solver is to be withdrawn, all registered model schemas which use that solver as their default should have this reference removed. The default solver for the DecisionNet AMPL subsystem should then be substituted for the solver to be withdrawn. If such model schemas are not updated, then system errors would result when they are executed.
- If a registered AMPL model schema is to be withdrawn, all registered datasets which use that model schema as their default should have this reference removed. If such datasets are not updated, then system errors may result when they are executed. In any event, the consumer will have to select a registered model schema known to be compatible with such a dataset.
- Because these business rules may prevent or dissuade an AMPL provider from withdrawing an object, they must be applied before the object is actually withdrawn. Therefore, the AMPL agent must apply the rules immediately after a provider communicates a desire to withdraw an AMPL object.

When a provider chooses to withdraw any DecisionNet object, the DecisionNet agent checks its database's TECHNOLO.DB table to determine whether the object to be withdrawn is independent or exclusive. If the object is independent, DecisionNet withdraws it immediately. If it is an AMPL object, the DecisionNet agent's module *WDTECHA* passes the object's unique identifier to the AMPL agent's module

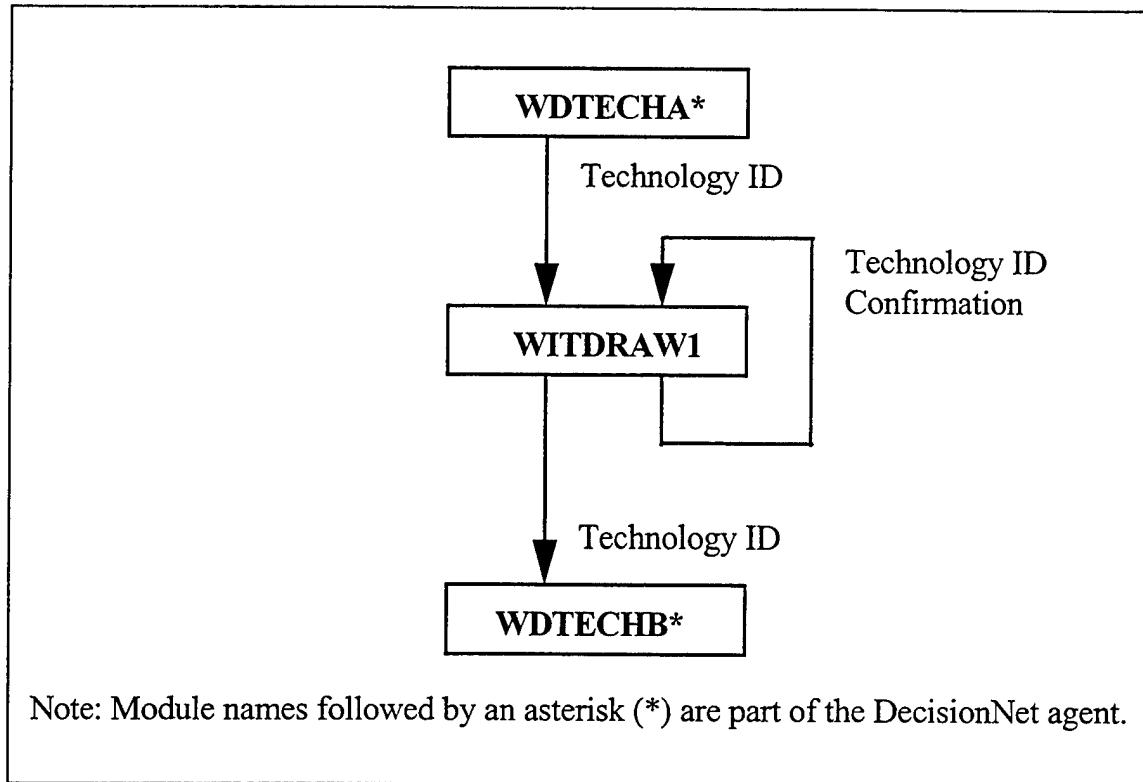


Figure 4. AMPL Object Withdrawal Process

WITDRAW1 module. *WITDRAW1* applies the three rules described above. If the object to be withdrawn is the AMPL default solver, it displays an appropriate message for the consumer and terminates the withdrawal process. If the object to be withdrawn is a solver with dependent model schemas, *WITDRAW1* lists those model schemas and requires the solver provider to confirm the withdrawal before continuing. If the object to be withdrawn is a model schema with dependent datasets, *WITDRAW1* lists those datasets and requires the model schema provider to confirm the withdrawal before continuing.

Once the business rules are *satisfied*, *WITDRAW1* deletes the object's entry from the appropriate table in the AMPL database. If the situation required confirmation by the provider to satisfy the business rules, then *WITDRAW1* also removes all other references to the object to be withdrawn. Finally, *WITDRAW1* passes the unique identifier of the withdrawn object to DecisionNet's module *WDTTECHB*, which deletes the object from its database table TECHNOLO.DB.

IV. IMPLEMENTATION NOTES

This section describes the process of transforming the DecisionNet AMPL project from a set of design specifications to a working product.

A prototype for this project was written in Borland Delphi® 1.0, using one program for each module in the AMPL object registration, update, withdrawal, and execution processes. Each program has its own initialization file (a short data file in human-readable text format) that contains the URL of the next program(s) to be executed. This would allow the programs to be moved, individually or as groups, to other directories or other server machines without recompiling any of the programs. This structure would ensure maximum flexibility for future improvements and expansions to DecisionNet.

Many of the programs in this project would require direct, read-only access to the DecisionNet database. The AMPL agent could obtain this access by placing all of its programs on the server machine that contains the DecisionNet database. The AMPL agent could also obtain access from any host machine networked with the server containing the DecisionNet database, if suitable networking protocols were in place.

A. AMPL DATABASE

The AMPL database for this prototype is accessed using Borland's IDAPI database engine. The database consists of five tables: MODEL.DB, DATASET.DB, SOLVER.DB, TMODEL.DB, and TDATASET.DB. Appendix A contains a data dictionary that describes each attribute.

Each table in the database represents a different type of AMPL object. The first three, MODEL, DATASET, and SOLVER, store AMPL objects registered with DecisionNet. The last two, TMODEL and TDATASET, store objects that are provided temporarily by DecisionNet consumers who wish to use an unregistered model schema and/or dataset on a single-use basis.

Each of the five tables uses an "autoincrement" field as its primary key. This type

of field is a sequential integer which holds no actual data on the record. It prevents key violations that might otherwise arise from attempting to reuse a key field value. The three permanent object tables also have a secondary key consisting of the provider ID and technology ID for the object stored. These values would be assigned by DecisionNet when an object is registered with DecisionNet. The AMPL agent would use these values to track the relationships between compatible model schemas, datasets, and solvers, and also to extract data from the DecisionNet database. The tables for temporary model schemas and datasets have no secondary key, as the objects they represent would not be registered with DecisionNet.

Each table in the database has a “Language” field, which contains the value “AMPL”. While this data is redundant, it would provide the means to merge the AMPL database with one or more databases from other exclusive DecisionNet technologies later.

B. AMPL PROVIDER SCRIPTS

1. AMPLREG

This program is the entry point for registering an AMPL object. It verifies that the ProviderID is active, and updates the DecisionNet active provider table. It then verifies that the DecisionNet registration process has begun for the given TechID. It generates an appropriate error message if the provider is not active, if the object is not listed with DecisionNet, or if the object type is not valid for AMPL. Finally, it hands control over to the appropriate follow-on program: *ASOLVREG* for a solver, *ADSREG* for a dataset, or *AMODREG* for a model. Its inputs are ProviderID and TechID. Its outputs are ProviderID and TechID.

2. ASOLVREG

This program lists the various requirements for an AMPL solver, and asks the user whether he wishes to continue the solver registration process or cancel (and return to the DecisionNet provider menu). HTML links to more in-depth information on the solver requirements are provided. If the user wants to register the solver, he or she must provide a URL for it. Its inputs are ProviderID and TechID. Its outputs are ProviderID, TechID,

YesNo, and URL.

3. ASLVREG2

This program either creates a new record in the AMPL solver table or returns the user to the DecisionNet registered provider menu, depending on the value of YesNo. If a solver is actually being registered, the next program called will complete the final portion of the DecisionNet registration process. Its inputs are ProviderID, TechID, YesNo, and URL. Its outputs are ProviderID and TechID.

4. ADSREG

This program is called by *AMPLREG* to register an AMPL dataset. It prompts the user to enter the dataset text and output script, and provides a current list of registered AMPL solvers for the user to choose from. This program obtains the list of registered AMPL solvers by querying the DecisionNet database directly. Each model schema is identified by its ProviderID and TechID, but the HTML "SELECT" structure allows only one parameter to be passed for each option. To resolve this limitation, the program concatenates the ProviderID and TechID, separated by a hyphen. This combined parameter is then called ModelID. This program's inputs are ProviderID, TechID. Its outputs are ProviderID, TechID, Dataset, and ModelID.

5. ADSREG2

This program adds the dataset text and default AMPL model to the Model table. To determine which model schema the user chose as default, the program breaks up the ModelID argument to find the appropriate ProviderID and TechID. It then passes control to *TECHDONE* to complete the DecisionNet portion of the registration process. Its inputs are ProviderID, TechID, Dataset, and ModelID. Its outputs are ProviderID and TechID.

6. AMODREG

This program prompts the user to enter the model text and output script, and provides a current list of registered AMPL solvers for the user to choose from. It obtains the list of registered AMPL solvers by querying the DecisionNet database directly. As in the *ADSREG* module, the program concatenates the ProviderID and TechID, separated by a hyphen. This combined parameter is then called SolverID. This program's inputs are

ProviderID and TechID. Its outputs are ProviderID, TechID, Model, Output, and SolverID.

7. AMODREG2

This program adds the model text, output script, and default AMPL solver to the Model table. To determine which solver the user chose as default, the program breaks up the SolverID argument to find the appropriate ProviderID and TechID. Control is then passed to *TECHDONE* to complete the DecisionNet portion of the registration process. This program's inputs are ProviderID, TechID, Model, Output, and SolverID. Its outputs are ProviderID and TechID.

8. UPDATE1

This program first verifies that the object type of the chosen technology has not been changed. It does this by matching the object type listed in the DecisionNet database with a record from the corresponding table in the AMPL agent database.

The program then generates a HTML form that displays the data fields for the AMPL object and allows the provider to update them. Since each type of AMPL object has different metainformation, the HTML form generated is specific to the object type. For model schemas, it lists all solvers in the DecisionNet database as default solver choices. For datasets, it lists all model schemas in the DecisionNet database as default model schema choices.

This program's inputs are ProviderID and TechID. Its outputs are ProviderID, TechID, SolverURL, Model, Output, SolverID, Dataset, and ModelID.

9. UPDATE2

This program stores the updated fields in the appropriate AMPL agent database table, which it first determines based on the object type listed in the DecisionNet database. Solver records are updated using a simple SQL "UPDATE" command. Model schema and dataset tables contain memo fields which are not supported by SQL, so for these object types the program deletes the existing record then inserts the new record. Finally, it sends the user back to DecisionNet's registered provider menu. Its inputs are ProviderID, TechID, SolverURL, Model, Output, SolverID, Dataset, and ModelID. Its

output is ProviderID.

10. WITDRAW1

This program is called from DecisionNet when the user wishes to withdraw an AMPL object. For solvers, it lists all registered models using that solver as default. For models, it lists all datasets using that model as default. In both cases, the data is drawn from the AMPL agent database. If any other AMPL objects use the selected object as default, this program requests verification via an HTML form that sends the user back to this same program again. The HTML form field Confirmed, which has a default value of false, tells the program whether or not the user has already confirmed the object withdrawal. If no other object uses the selected object as a default, or if the user has already confirmed withdrawal, this program deletes the object from the appropriate AMPL table, removes all references to it from other AMPL objects, and calls DecisionNet's program *WDTECHB* to delete all DecisionNet registration information on the object.

This program does not allow the user to withdraw the AMPL default solver, which is named in an initialization file on an accessible server. A system administrator must change to a different AMPL default solver (by modifying this initialization file) before the solver can be withdrawn.

This program's inputs are ProviderID, TechID, and Confirmed. Its outputs are ProviderID, TechID, and Confirmed.

C. AMPL CONSUMER SCRIPTS

1. AMPLEXEC

This program is called from DecisionNet when the user wishes to execute an AMPL object (either a model schema, dataset, or solver). It verifies that the UserID is an active consumer or provider, and that the chosen technology is a registered AMPL dataset, model schema, or solver. It then passes the ProviderID and TechID of the chosen object to the appropriate program for that object type: *AMPLEXE2* for model schemas, *AMPLEXE3* for datasets, and *AMPLEXE4* for solvers. Its inputs are UserID, ProviderID,

TechID. Its outputs are ProviderID, and TechID.

2. AMPLEXE2

This program is called by *AMPLEXEC* when a consumer wishes to execute an AMPL model schema. The user is prompted to either select a registered dataset (from list) or enter his own dataset. The user's response is processed by the next program called, *AMPLEXE5*. Its inputs are ProviderID and TechID. Its outputs are Method, DatasetID, Dataset, ModelProvID, ModelTechID, and SolverURL.

3. AMPLEXE3

This program is called by *AMPLEXEC* when a consumer wishes to execute an AMPL dataset. The user is prompted to either select the default model, select another registered model (from list) or enter his own model. The user's response is processed by the next program called, *AMPLEXE6*. Its inputs are ProviderID, TechID, and SolverURL. Its outputs are DefaultProvID, DefaultTechID, Method, ModelID, Model, Output, SolverID, SolverURL, DSProvID, and DSTechID.

4. AMPLEXE4

This program is called by *AMPLEXEC* when a consumer wishes to use an AMPL solver. The user is prompted to either select a registered model schema (from list), select a registered dataset (from list) or enter his own model schema and dataset. The user's response is processed by the next program, *AMPLEXE7*. The inputs to *AMPLEXE4* are ProviderID and TechID. Its outputs are Method, SolverURL, ModelID, DatasetID, Model, Dataset, and Output.

5. AMPLEXE5

This program processes the user responses requested by *AMPLEXE2*. It determines whether the user wishes to use a registered dataset or provide his own. If an unregistered dataset is provided, it is stored temporarily in the TDataset table, and its DatasetNumber is prefixed with a "t". This program also looks up the default solver for the chosen model schema. The inputs to *AMPLEXE5* are Method, DatasetID, Dataset, ModelProvID, ModelTechID, and SolverURL. Its outputs are ModelNumber, DatasetNumber, and SolverURL.

6. **AMPLEXE6**

This program processes the user responses requested by *AMPLEXE3*. It determines whether the user wishes to use a registered model schema or provide his own. If an unregistered model schema is provided, it is stored temporarily in the TModel table, and its ModelNumber is prefixed with a "T". This program also looks up the default solver for the chosen model schema. Its inputs are Method, ModelID, DefaultProvID, DefaultTechID, SolverID, SolverURL, Model, Output, DSProvID, and DSTechID. Its outputs are ModelNumber, DatasetNumber, and SolverURL.

7. **AMPLEXE7**

This program processes the user responses requested by *AMPLEXE4*. It determines whether the user wishes to use a registered model schema or dataset, or provide his own. If an unregistered model schema and dataset are provided, they are stored temporarily in the TModel and TDataset tables and *AMPLEXE9* is called. If a registered model schema is chosen, it calls *AMPLEXE2*. If a registered dataset is chosen, it calls *AMPLEXE3*. Its inputs are Method, SolverURL, ModelID, DatasetID, Model, Dataset, and Output. Its outputs are ModelNumber, DatasetNumber, SolverURL, ProviderID, and TechID.

8. **AMPLEXE9**

This program allows the user to verify the model schema, dataset, and execution script just before they are sent to the AMPL solver. It retrieves the execution script from either the Model or TModel table.

AMPLEXE9 also generates a unique, extensionless filename (i.e., a filename stub) for each request it processes. The filename stub provides a means to differentiate between multiple AMPL output files generated from concurrent requests being handled on a single AMPL Agent host.

When the user submits the form generated by this program, the AMPL solver agent residing at the SolverURL is called. The input to *AMPLEXE9* are ModelNumber, DatasetNumber, and SolverURL. Its outputs are Model, Dataset, Output, and FileStub.

D. AMPL SOLVER AGENT SCRIPT

The *AMPLCALL* program is executed by the AMPL Agent's *AMPLEXE9* program to invoke an AMPL solver agent and supply it with a model schema, dataset, and output script. It receives these inputs from three separate HTML text areas passed when *AMPLCALL* is invoked. It also receives the unique filename stub generated by *AMPLEXE9*.

The model schema is saved to the solver agent machine's local disk drive under the temporary filename, with a ".MOD" file extension. Likewise, the dataset and output script are also saved, using ".DAT" and ".RUN" file extensions, respectively. Since only one AMPL script file can be passed when invoking the AMPL program, the solver agent must augment the output script (i.e., the ".RUN" file) with references to the model schema and dataset. The resulting AMPL script file begins with references to the ".MOD" file and the ".DAT" file, followed by the actual output script.

To properly capture and format the output of the AMPL program, *AMPLCALL* creates a customized DOS batch file. This file includes HTML tags that make the resultant output file readable as HTML text. It also includes a command that captures the AMPL output and saves it using the filename stub received from the *AMPLRUN* module, with an ".HTM" extension. The DOS batch file itself is named using the temporary filename generated as described above, with a ".BAT" extension.

AMPLCALL then invokes a DOS command shell to execute the DOS batch file. The batch file invokes AMPL and sends its output sent to the ".HTM" file in HTML format.

The Solver Agent then sends the output file back to the AMPL Agent host using the ftp upload protocol. In order to do this, the Solver Agent must hold an ftp user identifier and accompanying password, as well as the URL for the AMPL Agent's ftp server. This information is stored in *AMPLCALL*'s ".INI" file, so it may be examined and modified without revising or recompiling the *AMPLCALL* program.

Finally, *AMPLCALL* sends the user an HTML file that states an AMPL output file

has been generated, and provides an HTML link to the output file on the AMPL Agent host. The WWW server URL for the AMPL Agent host is also stored in *AMPLCALL*'s ".INT" file.

V. CONCLUSIONS AND RECOMMENDATIONS

A. AMPL AS AN EXCLUSIVE TECHNOLOGY FOR DecisionNet

The addition of AMPL to DecisionNet demonstrates the concept of exclusive technologies. This thesis describes a conceptual framework that can be applied to exclusive technologies in general, and provides a database structure capable of supporting them. A DecisionNet AMPL agent would provide three new benefits to its users.

1. Searchable Repository of Exclusive Objects

Each model schema, dataset, and solver in the AMPL agent database is also catalogued in the DecisionNet database. This gives consumers the ability to search for AMPL objects based on: object type, problem area, functional area, industry type, and organization type. DecisionNet also gives users a short description of each registered technology, and the ability to search its database based on keywords. [1]

After choosing an AMPL object from the DecisionNet database, the consumer could create a complete AMPL problem by choosing complementary AMPL objects in the same manner. Alternately, the AMPL agent could provide the required complementary objects to complete the AMPL problem. If the consumer first chooses a model schema, the AMPL agent will suggest an appropriate solver. If the consumer first chooses a dataset, the AMPL agent will suggest an appropriate model schema and solver.

The ability to search the DecisionNet database, combined with the ability of the a AMPL agent to link together compatible AMPL objects, create a mathematical modeling environment that promotes reuse and sharing of computational objects.

2. Mathematical Modeling for New Audiences

A DecisionNet AMPL agent would provide a means for anyone with WWW access to formulate and execute AMPL problems. This agent would require little or no knowledge of the AMPL language from consumers who use compatible sets of registered model schemas and datasets. It would also allow consumers to easily modify registered model schemas and datasets to meet their particular needs.

Consumers who have experience in AMPL could also reap new benefits from a DecisionNet AMPL agent. The ability to choose from a variety of registered solvers on providers' host platforms could enable such consumers to solve problems more complex than would be possible with a typical standalone AMPL environment.

3. Provider Control of Registered Objects

Traditionally, providers of decision support technologies relinquish physical control of their products once they are given to users, and must then rely on software licensing agreements to control their usage. By registering their exclusive technologies with DecisionNet, providers would gain the ability to modify or withdraw their registered exclusive objects at any time.

A provider could also limit the extent to which a solver is used, and could even restrict the use of a solver to a specific group of consumers.

These added controls made possible by the DecisionNet agent would substantially increase the potential for commercial value to providers.

B. SUGGESTIONS FOR FURTHER DEVELOPMENT

1. Additional Exclusive Technologies

This project provides a core database and structure for incorporating exclusive technologies into DecisionNet. The AMPL agent database is designed to accommodate objects from other exclusive technologies via the "Language" field in every table. The contents of this field identify the exclusive technology for which each object was registered.

Assume, for instance, that GAMS is to be added to DecisionNet as an exclusive technology [2]. Like AMPL, GAMS requires a model schema, data, and a solver. A DecisionNet GAMS agent could register objects in the exclusive database already used by the AMPL agent. When a user searches the database for a model schema, both AMPL and GAMS objects would be listed. This integration would alleviate the need to arbitrarily choose an exclusive technology to solve a particular problem. Instead, the user

could choose based solely on the suitability of the objects themselves. [7]

In the case of AMPL and GAMS, a dataset could even be dually registered to make it available for use with model schemas of either language. The resulting freedom of choice would greatly increase the applicability of the dataset by allowing the user choose from more model schemas.

2. Graphically-Oriented Exclusive Technologies

AMPL, as implemented in DecisionNet, is solely text-based in its ability to display model schemas, data, and output. Many other decision support software tools provide graphically oriented representations of problems and solutions. Since the WWW uses a graphical user interface, a graphically oriented exclusive technology could be added to DecisionNet in exactly the same manner as AMPL.

3. Saving State

The AMPL agent scripts save the state of a user session by passing state data through hidden HTML form fields from one script to the next. While this method is functional, it does not provide much security. Newer WWW developments may provide a more elegant and secure method to save the state of AMPL agent user sessions.

4. Electronic Commerce

As DecisionNet grows in size and popularity, the value of its collection of decision support objects will increase. However, the providers of exclusive technologies such as solvers presently incur overhead costs by making their own server platforms available to the public. There is currently no financial incentive for technology providers to register decision support objects with DecisionNet.

Implementation of an electronic commerce system on DecisionNet would provide that incentive by charging users a small fee for accessing selected registered objects, and paying the technology providers a portion of the revenues.

APPENDIX A. DATA DICTIONARY

Attribute	Description	Format	Tables	Key Field	Required
ModelNumber	Incrementally assigned model schema serial number	Integer	MODEL.DB TMODEL.DB	Primary	Yes
DatasetNumber	Incrementally assigned dataset serial number	Integer	DATASET.DB TDATASET.DB	Primary	Yes
SolverNumber	Incrementally assigned solver serial number	Integer	SOLVER.DB	Primary	Yes
ProviderID	Identifies the DecisionNet provider who registers an object	1-15 Characters	MODEL.DB DATASET.DB SOLVER.DB	Secondary	Yes
TechID	Distinguishes a particular object from others registered by the same provider	1-3 Characters	MODEL.DB DATASET.DB SOLVER.DB	Secondary	Yes
Language	Identifies the Exclusive Technology to be used (example: AMPL)	1-10 Characters	ALL	No	Yes
Model	Contains the actual text of a model schema	ASCII Text	MODEL.DB TMODEL.DB	No	No
Output	Contains the actual text of an output script	ASCII Text	MODEL.DB TMODEL.DB	No	No
Dataset	Contains the actual text of a dataset	ASCII Text	DATASET.DB TDATASET.DB	No	No

SolverURL	Full WWW address of a solver host	ASCII Text	SOLVER.DB	No	No
ModelProvID	ProviderID of the default model for a dataset	1-15 Characters	DATASET.DB	No	No
ModelTechID	TechID of the default model for a dataset	1-3 Characters	DATASET.DB	No	No
SolverProvID	ProviderID of the default solver for a model	1-15 Characters	MODEL.DB	No	No
SolverTechID	TechID of the default solver for a model	1-3 Characters	MODEL.DB	No	No
Created	Date and Time object is created	Date/Time	TMODEL.DB TDATASET.DB	No	No

APPENDIX B. SOURCE CODE

This appendix contains Delphi source code for all program modules described in this thesis.

The author chose Delphi for this project because its IDAPI database engine is also used for DecisionNet. This gives the AMPL agent transparent access to the DecisionNet database. Delphi's executable code is extremely portable among Windows platforms, and its third party support for WWW scripting components is excellent.

AMPLREG

```
unit Amplreg1;

{ This program is the entry point for registering an AMPL object.
  It verifies that the ProviderID is active, and updates the active
  provider table (for DecisionNet). It then verifies that the Dnet
  registration process has begun for the given TechID. Finally, it
  hands control over to the appropriate follow-on program: ASOLVREG for
  a solver, ADSREG for a dataset, or AMODREG for a model.
  INPUTS: ProviderID, TechID
  OUTPUTS: ProviderID, TechID
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, StdCtrls, Toolbar, ExtCtrls,
  UpdateOk, TpShell, IniLink, Ftp, starsock, DB, DBTables, Grids,
  DBGrids;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    IniLink: TIniFileLink;
    TechTempTable: TTable;
    ActProvTable: TTable;
    DataSource1: TDataSource;
    DBGrid1: TDBGrid;
    procedure ProcessHtmlForm(Sender: TObject);
    procedure SendHtmlHeader;
    procedure SendHtmlFooter;
    procedure UpdateActiveProvider;
    procedure CreateForm;

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

  ProviderID : String;
  TechID      : String;
  ObjectType  : String;

implementation

{$R *.DFM}

procedure TForm1.ProcessHtmlForm(Sender: TObject);
begin
  with CGIEnvData1 do
    begin
      {required when this program runs under WebSite}
      webSiteINIFilename := paramstr(1);
      application.OnException := cgiErrorHandler;
    end;
  end;
end;
```

```

application.processMessages;

{send header for cgi output to client}
SendHtmlHeader;

{Read in ProviderID and TechID from form literals}
ProviderID := GetSmallField('ProviderID');
TechID := GetSmallField('TechID');

{Check Active Provider table and update time of last action}
UpdateActiveProvider;

{Get object type from TechTemp table}
with TechTempTable do
begin
  Open;
  SetKey;
  FieldByName('ProviderID').AsString := ProviderID;
  FieldByName('TechID').AsString := TechID;
  If not GotoKey then {Technology not registered with DNet}
  begin
    sendHdr('3','This technology must first be registered with
    DecisionNet. ');
    send('Please go back to DecisionNet and try again. ');
    SendHTMLFooter;
  end
  else {Get object type}
  ObjectType := FieldByName('tObjectType').AsString;
end;

{Go to appropriate .exe for the type of object being registered}
If (ObjectType = 'Dataset') or (ObjectType = 'Model Schema') or
  (ObjectType = 'Solver') then {ObjectType is a valid AMPL object}
  CreateForm
else
begin
  TechTempTable.delete;
  sendHdr('3','" + ObjectType + '" is not an allowed AMPL
object. ');
  send('Allowed objects include: Model Schema, Dataset, and
Solver.<BR> ');
  send('Please go back to DecisionNet and try again. ');
  SendHTMLFooter;
end;

end;
end;

procedure TForm1.CreateForm;
begin
  with CGIEnvData1 do
  begin
    send('Please press the button to begin registering your AMPL ');
    send( ObjectType + '. ');
    if ObjectType = 'Dataset' then send('<form method=post action="' +
      IniLink.StringEntry['ExePath'] +
      IniLink.StringEntry['DatasetExe']);
    if ObjectType = 'Model Schema' then send('<form method=post
action="' + IniLink.StringEntry['ExePath'] +
      IniLink.StringEntry['ModelExe'] );
  end;
end;

```

```

        if ObjectType = 'Solver' then send('<form method=post action="' +
            IniLink.StringEntry['ExePath'] + IniLink.StringEntry['SolverExe']
    );
    send('"><BR>');
    send('<INPUT TYPE=HIDDEN NAME="ProviderID" VALUE="' + ProviderID +
        '">');
    send('<INPUT TYPE=HIDDEN NAME="TechID" VALUE="' + TechID + '">');
    send(' <input type=submit value="Continue"></form></CENTER>' );
end;

SendHTMLFooter;
end;
procedure TForm1.SendHtmlHeader;
begin
    with CGIEnvData1 do
    begin
        {HTML page header info}
        createStdout;
        sendPrologue;

        send( '<HTML><HEAD>' );
        sendTitle( 'DecisionNet AMPL Object Registration' );
        send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
        sendHdr( '2','DecisionNet AMPL Object Registration' );
        sendHR;

    end;
end;

procedure TForm1.SendHtmlFooter;
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}
        send('<HR>');
        send( 'This application was created by Michael Casey for Professor
            ' );
        send( 'Hemant Bhargava.<br>' );
        send( 'Generated on ' + webdate(now) );
        send( '</BODY></HTML>' );
        closeStdout;

        {quit application}
        closeApp(application);
    end;
end;

procedure TForm1.UpdateActiveProvider;
begin
    {Check Active Provider table and update time of last action}
    with ActProvTable, CGIEnvData1 do
    begin
        Open;
        If not FindKey([ProviderID]) then {Provider is not logged in}
        begin
            sendHdr('3',ProviderID + ' is not logged in. ');
            send('Please log in to DecisionNet and try again. ');
            SendHTMLFooter;
        end
    end
end;

```

```
else {Update date and time of last action}
begin
  Edit;
  FieldByName('LastActionTime').AsString := DateTimetoStr(now);
end;
Close;
end;
end;
end.
```

ASOLVREG

```
unit Asolreg1;

{ This program lists the various requirements for an AMPL solver, and
asks the user whether he wishes to continue the solver registration
process or cancel. HTML links to more in-depth information on the solver
requirements are provided. The user's decision is processed by the next
program, which is ASLVREG2. If the user indeed wants to register the
solver, he must also provide a URL for it.
  INPUTS: ProviderID, TechID
  OUTPUTS: ProviderID, TechID, YesNo, URL
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, StdCtrls, Toolbar, ExtCtrls,
  UpdateOk, TpShell, IniLink, Ftp, starsock, DB, DBTables, Grids,
  DBGrids;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    TechTempTable: TTable;
    DataSource1: TDataSource;
    ActProvTable: TTable;
    procedure ProcessHtmlForm(Sender: TObject);
    procedure SendHtmlHeader;
    procedure SendHtmlFooter;
    procedure UpdateActiveProvider;
    procedure CreateForm;

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

  ProviderID : String;
  TechID      : String;

implementation

{$R *.DFM}

procedure TForm1.ProcessHtmlForm(Sender: TObject);
begin
  with CGIEnvData1 do
    begin
      {required when this program runs under WebSite}
      webSiteINIFilename := paramstr(1);
      application.OnException := cgiErrorHandler;
    end;
  end;
end;
```

```

application.processMessages;

{send header for cgi output to client}
SendHtmlHeader;

{Read in ProviderID and TechID from form literals}
ProviderID := GetSmallField('ProviderID');
TechID := GetSmallField('TechID');

{Check Active Provider table and update time of last action}
UpdateActiveProvider;

{Send HTML form}
CreateForm;
end;
end;

procedure TForm1.CreateForm;
begin
  with CGIEnvData1 do
    begin
      sendHdr('4','You have chosen to register an AMPL solver with
DecisionNet');
      send(' </CENTER>');
      send('A DecisionNet AMPL solver requires the following
items.<BR><BR>');
      send('Hardware:');
      send(' <UL><LI>A server with continuous Internet access');
      send(' <LI>A static IP address</UL>');
      send('Software:');
      send(' <UL><LI>A ');
      send(' <a href="http://www.comvista.com/us/lea/servers.html">WWW
server</A>');
      send(' application');
      send(' <LI>A copy of the ');
      send(' <A
HREF="http://www.ampl.com/cm/cs/what/ampl/index.html">AMPL</A>');
      send(' application');
      send(' <LI>One or more ');
      send(' <a href="http://www.ampl.com/cm/cs/what/ampl/solvers.html">');
      send(' solver algorithms</a> linked to the AMPL application');
      send(' <LI>A copy of the DecisionNet ');
      send(' <a href="' + IniLink.StringEntry['SolverAgentPage'] + '>' +
'AMPL Solver Agent</a> application');
      send(' </UL><HR>');
      send(' <form method=post action="' + IniLink.StringEntry['ExePath'] +
IniLink.StringEntry['NextExe'] + '><BR>');
      send(' <input type=radio name="YesNo" value="Yes">');
      send(' <B>Yes, I have all required hardware and software, and am ' +
'ready to register my AMPL solver with DecisionNet.<center>');
      send(' <br>The URL is </B><input type=text name="URL" length=255
size=50 ' +
'value="http://"></CENTER><BR><BR>');
      sendHR;
      send(' <BR><input type=radio name="YesNo" value="No" selected>');
      send(' <B>No, I am not ready to ' +
'register my AMPL solver with DecisionNet at this
time.</B><BR><BR>');
      send(' <HR>');
      send(' <input type=hidden name="ProviderID" value="' + ProviderID +

```

```

        '>');
        send('<input type=hidden name="TechID" value="' + TechID + '>');
        send( '<CENTER><input type=submit value="Continue"></CENTER></form>'
        );
    end;

    SendHTMLFooter;
end;

procedure TForm1.SendHtmlHeader;
begin
    with CGIEnvData1 do
    begin
        {HTML page header info}
        createStdout;
        sendPrologue;

        send( '<HTML><HEAD>' );
        sendTitle( 'DecisionNet AMPL Solver Registration' );
        send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
        sendHdr( '2','DecisionNet AMPL Solver Registration' );
        sendHR;

    end;
end;

procedure TForm1.SendHtmlFooter;
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}
        send( 'This application was created by Michael Casey for Professor
        ');
        send( 'Hemant Bhargava.<br>' );
        send( 'Generated on ' + webdate(now) );
        send( '</BODY></HTML>' );
        closeStdout;

        {quit application}
        closeApp(application);
    end;
end;

procedure TForm1.UpdateActiveProvider;
begin
    {Check Active Provider table and update time of last action}
    with ActProvTable, CGIEnvData1 do
    begin
        Open;
        If not FindKey([ProviderID]) then {Provider is not logged in}
        begin
            sendHdr('3',ProviderID + ' is not logged in. ');
            send('Please log in to DecisionNet and try again. ');
            SendHTMLFooter;
        end
        else {Update date and time of last action}
        begin
            Edit;

```



```
        FieldByName('LastActionTime').AsString := DateTimetoStr(now);
    end;
    Close;
end;
end;
end.
```

ASLVREG2

```
unit Asolreg2;

{ This program processes the user output from ASOLVREG and either
updates the AMPL solver table or returns the user to the Dnet registered
provider menu, as appropriate. If a solver is actually being registered,
the next program called will complete the final Dnet portion of the
registration process.
  INPUTS: ProviderID, TechID, YesNo, URL
  OUTPUTS: ProviderID, TechID
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, StdCtrls, Toolbar, ExtCtrls,
  UpdateOk, TpShell, IniLink, Ftp, starsock, DB, DBTables, Grids,
  DBGrids;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    TblSolver: TTable;
    procedure ProcessHtmlForm(Sender: TObject);
    procedure SendHtmlHeader;
    procedure SendHtmlFooter;
    procedure CreateForm;

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

  YesNo : String;
  URL   : String;
  ProviderID, TechID : String;

implementation

{$R *.DFM}

procedure TForm1.ProcessHtmlForm(Sender: TObject);
begin
  with CGIEnvData1 do
  begin
    {required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.OnException := cgiErrorHandler;
    application.ProcessMessages;

    {send header for cgi output to client}
  end;
end;
```

```

    SendHtmlHeader;

    {Read in ProviderID and TechID from form literals}
    ProviderID := GetSmallField('ProviderID');
    TechID := GetSmallField('TechID');
    YesNo := GetSmallField('YesNo');
    URL := GetSmallField('URL');

    {Send HTML form}
    CreateForm;
end;
end;

procedure TForm1.CreateForm;
begin
    with CGIEnvData1 do
    begin
        if YesNo = 'Yes' then {register solver}
        begin
            with TblSolver do
            begin
                open;
                AppendRecord(['', ProviderID, TechID, 'AMPL', URL]);
                close;
            end;
            send('<CENTER>');
            sendHdr('4', 'Please press the button to complete the solver ' +
                'registration process. ');
            send('<BR><BR><form method=post action="" +
IniLink.StringEntry['ExePath'] +
                IniLink.StringEntry['NextExe'] + '>');
            send('<input type=hidden name="ProviderID" value="" + ProviderID +
                ">');
            send('<input type=hidden name="TechID" value="" + TechID + ">');
            send(' <CENTER><input type=submit
value="Continue"></CENTER></form>'
                );
            SendHtmlFooter;
        end
        else {send user back to Registered Provider Menu}
        begin
            send('<CENTER>');
            sendHdr('4', 'Your solver registration has been cancelled. ');
            send('Please press the button to return to the DecisionNet
registered provider menu. ');
            send('<BR><BR><form method=post action="" +
IniLink.StringEntry['ExePath'] +
                IniLink.StringEntry['MenuExe'] + '>');
            send('<input type=hidden name="ProviderID" value="" + ProviderID +
                ">');
            send(' <CENTER><input type=submit
value="Continue"></CENTER></form>'
                );
            SendHtmlFooter;
        end;
    end;
end;

procedure TForm1.SendHtmlHeader;
begin
    with CGIEnvData1 do
    begin

```

```

    {HTML page header info}
    createStdout;
    sendPrologue;

    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Solver Registration' );
    send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
    sendHdr( '2','DecisionNet AMPL Solver Registration' );
    sendHR;

end;
end;

procedure TForm1.SendHtmlFooter;
begin
    with CGIEnvData1 do
        begin
            {HTML page footer info}
            send( 'This application was created by Michael Casey for Professor
                ' );
            send( 'Hemant Bhargava.<br>' );
            send( 'Generated on ' + webdate(now) );
            send( '</BODY></HTML>' );
            closeStdout;

            {quit application}
            closeApp(application);
        end;
    end;
end;
end.

```

ADSREG

```
unit Adsreg1;
```

```
{ This program is called by AMPLREG to register an AMPL dataset. It
prompts the user to enter the dataset text and output script, and
provides a current list of registered AMPL solvers for the user to
choose from. This data is processed by ADSREG2.EXE.
```

```
  INPUTS: ProviderID, TechID
```

```
  OUTPUTS: ProviderID, TechID, dataset, ModelID
```

```
}
```

```
interface
```

```
uses
```

```
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, StdCtrls, Toolbar, ExtCtrls,
  UpdateOk, TpShell, IniLink, Ftp, starsock, DB, DBTables, Grids,
  DBGrids;
```

```
type
```

```
  TForm1 = class(TForm)
```

```
    CGIEnvData1: TCGIEnvData;
```

```
    TpStatusBar1: TtpStatusBar;
```

```
    TpStatusPanel1: TtpStatusPanel;
```

```
    IniLink: TIniFileLink;
```

```
    TechTempTable: TTable;
```

```
    ActProvTable: TTable;
```

```
    QueryModels: TQuery;
```

```
    procedure ProcessHtmlForm(Sender: TObject);
```

```
    procedure SendHtmlHeader;
```

```
    procedure SendHtmlFooter;
```

```
    procedure UpdateActiveProvider;
```

```
    procedure CreateForm;
```

```
  private
```

```
    { Private declarations }
```

```
  public
```

```
    { Public declarations }
```

```
end;
```

```
var
```

```
  Form1: TForm1;
```

```
  ProviderID : String;
```

```
  TechID      : String;
```

```
implementation
```

```
{ $R *.DFM }
```

```
procedure TForm1.ProcessHtmlForm(Sender: TObject);
```

```
begin
```

```
  with CGIEnvData1 do
```

```
  begin
```

```
    {required when this program runs under WebSite}
```

```
    webSiteINIFilename := paramstr(1);
```

```
    application.OnException := cgiErrorHandler;
```

```
    application.ProcessMessages;
```

```

    {send header for cgi output to client}
    SendHtmlHeader;
    {Read in ProviderID and TechID from form literals}
    ProviderID := GetSmallField('ProviderID');
    TechID := GetSmallField('TechID');

    {Check Active Provider table and update time of last action}
    UpdateActiveProvider;

    {Send HTML form}
    CreateForm;
end;
end;

procedure TForm1.CreateForm;
begin
    with CGIEnvData1 do
    begin
        sendHdr('4','You have chosen to register an AMPL dataset with
DecisionNet. ');
        send( '</CENTER>');
        send('You will be asked to:');
        send('<UL><LI>Provide the text contents of the dataset');
        send('<LI>Specify a default model schema to be used with this
dataset');
        send('<form method=post action="' + IniLink.StringEntry['ExePath'] +
IniLink.StringEntry['NextExe'] + '"><BR>');
        SendHdr('4','1. Enter the dataset in the box below:<BR>');
        send( '<textarea name="dataset" rows=20 cols=80>' );
        send( '</textarea><BR><BR>' );
        sendHdr('4','2. Choose a default model from the list below:');
        Send('<SELECT NAME="ModelID" TYPE=TEXT>');
        send('<OPTION VALUE="none">No default model schema');
        with QueryModels do
        begin
            open;
            while not EOF do
            begin
                send('<OPTION VALUE="' + FieldByName('ProviderID').AsString +
                +
                FieldByName('TechID').AsString + '">');
                send(FieldByName('TechName').AsString + ' (' +
                FieldByName('ProviderID').AsString + '-' +
                FieldByName('TechID').AsString + ')');
            next;
        end;
    end;
    send('</SELECT><BR><BR>');

    send('<input type=hidden name="ProviderID" value="' + ProviderID +
    '">');
    send('<input type=hidden name="TechID" value="' + TechID + '">');
    send( '<CENTER><input type=submit value="Submit
form"></CENTER></form>' );
    end;
    SendHTMLFooter;
end;

procedure TForm1.SendHtmlHeader;
begin

```

```

with CGIEnvData1 do
begin
    {HTML page header info}
    createStdout;
    sendPrologue;
    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Dataset Registration' );
    send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
    sendHdr( '2','DecisionNet AMPL Dataset Registration' );
    sendHR;

    end;
end;

procedure TForm1.SendHtmlFooter;
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}
        sendHR;
        send( 'This application was created by Michael Casey for Professor
            ' );
        send( 'Hemant Bhargava.<br>' );
        send( 'Generated on ' + webdate(now) );
        send( '</BODY></HTML>' );
        closeStdout;

        {quit application}
        closeApp(application);
    end;
end;

procedure TForm1.UpdateActiveProvider;
begin
    {Check Active Provider table and update time of last action}
    with ActProvTable, CGIEnvData1 do
    begin
        Open;
        If not FindKey([ProviderID]) then {Provider is not logged in}
        begin
            sendHdr('3',ProviderID + ' is not logged in. ');
            send('Please log in to DecisionNet and try again. ');
            SendHTMLFooter;
        end
        else {Update date and time of last action}
        begin
            Edit;
            FieldByName('LastActionTime').AsString := DateTimetoStr(now);
        end;
        Close;
    end;
end;

end.

```

ADSREG2

```
unit Adsreg_2;
```

```
{ This program is called by ADSREG to register an AMPL dataset. It adds
the dataset text and default AMPL model to the Model table. Control is
then passed to TECHDONE to complete the Dnet portion of the registration
process.
```

```
  INPUTS: ProviderID, TechID, dataset ModelID
```

```
  OUTPUTS: ProviderID, TechID
```

```
}
```

```
interface
```

```
uses
```

```
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, StdCtrls, Toolbar, ExtCtrls,
  UpdateOk, TpShell, IniLink, Ftp, starsock, DB, DBTables, Grids,
  DBGrids;
```

```
type
```

```
  TForm1 = class(TForm)
```

```
    CGIEnvData1: TCGIEnvData;
```

```
    TpStatusBar1: TtpStatusBar;
```

```
    TpStatusPanel1: TtpStatusPanel;
```

```
    IniLink: TIniFileLink;
```

```
    ActProvTable: TTable;
```

```
    TblDataset: TTable;
```

```
    procedure ProcessHtmlForm(Sender: TObject);
```

```
    procedure SendHtmlHeader;
```

```
    procedure SendHtmlFooter;
```

```
    procedure UpdateActiveProvider;
```

```
    procedure CreateForm;
```

```
    procedure AddDataset;
```

```
  private
```

```
    { Private declarations }
```

```
  public
```

```
    { Public declarations }
```

```
end;
```

```
var
```

```
  Form1: TForm1;
```

```
  ProviderID    : String;
```

```
  TechID        : String;
```

```
  Dataset       : TStringList;
```

```
  ModelID       : String;
```

```
  ModelProvID   : String;
```

```
  ModelTechID   : String;
```

```
implementation
```

```
{ $R *.DFM }
```

```
procedure TForm1.ProcessHtmlForm(Sender: TObject);
```

```
begin
```

```
  with CGIEnvData1 do
```

```
  begin
```

```
    {required when this program runs under WebSite}
```



```

    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {send header for cgi output to client}
    SendHtmlHeader;

    {Read in ProviderID and TechID from form literals}
    ProviderID := GetSmallField('ProviderID');
    TechID := GetSmallField('TechID');

    {Check Active Provider table and update time of last action}
    UpdateActiveProvider;

    {Read in dataset text and default model choice}
    Dataset := TStringList.create;
    Dataset.clear;
    GetTextArea( 'dataset' , Dataset );

    ModelID := GetSmallField('ModelID');

    {Break ModelID into ModelProvID and ModelTechID}
    ModelProvID := Copy(ModelID, 0, Pos('/',ModelID)-1 );
    ModelTechID := Copy(ModelID, Pos('/',ModelID)+1 , 10);

    {Update database}
    AddDataset;

    {Send HTML form}
    CreateForm;
end;
end;

procedure TForm1.AddDataset;
begin
    with TblDataset do
    begin
        open;
        AppendRecord(['', ProviderID, TechID, 'AMPL', Dataset, ModelProvID,
            ModelTechID, 'Yes']);
        close;
    end;
end;

procedure TForm1.CreateForm;
begin
    with CGIEnvData1 do
    begin
        send('<CENTER>');
        sendHdr('4', 'Please press the button to complete the dataset ' +
            'registration process. ');
        send('<BR><BR><form method=post action="' +
            IniLink.StringEntry['ExePath'] +
            IniLink.StringEntry['NextExe'] + '">');
        send('<input type=hidden name="ProviderID" value="' + ProviderID +
            '">');
        send('<input type=hidden name="TechID" value="' + TechID + '">');
        send( '<CENTER><input type=submit value="Continue"></CENTER></form>'
            );
    end;
end;

```

```

    SendHTMLFooter;
end;

procedure TForm1.SendHtmlHeader;
begin
    with CGIEnvData1 do
    begin
        {HTML page header info}
        createStdout;
        sendPrologue;

        send( '<HTML><HEAD>' );
        sendTitle( 'DecisionNet AMPL Dataset Registration' );
        send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
        sendHdr( '2','DecisionNet AMPL Dataset Registration' );
        sendHR;

    end;
end;

procedure TForm1.SendHtmlFooter;
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}
        sendHR;
        send( 'This application was created by Michael Casey for Professor
        ');
        send( 'Hemant Bhargava.<br>' );
        send( 'Generated on ' + webdate(now) );
        send( '</BODY></HTML>' );
        closeStdout;

        {quit application}
        closeApp(application);
    end;
end;

procedure TForm1.UpdateActiveProvider;
begin
    {Check Active Provider table and update time of last action}
    with ActProvTable, CGIEnvData1 do
    begin
        Open;
        If not FindKey([ProviderID]) then {Provider is not logged in}
        begin
            sendHdr('3',ProviderID + ' is not logged in. ');
            send('Please log in to DecisionNet and try again. ');
            SendHTMLFooter;
        end
        else {Update date and time of last action}
        begin
            Edit;
            FieldByName('LastActionTime').AsString := DateTimetoStr(now);
        end;
        Close;
    end;
end;
end;

```

end.

AMODREG

```
unit Amodreg1;

{ This program is called by AMPLREG to register an AMPL model. It
  prompts the user to enter the model text and output script, and provides
  a current list of registered AMPL solvers for the user to choose from.
  This data is processed by AMODREG2.EXE.
  INPUTS: ProviderID, TechID
  OUTPUTS: ProviderID, TechID, model, output, SolverID
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, StdCtrls, Toolbar, ExtCtrls,
  UpdateOk, TpShell, IniLink, Ftp, starsock, DB, DBTables, Grids,
  DBGrids;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    TechTempTable: TTable;
    ActProvTable: TTable;
    QuerySolvers: TQuery;
    procedure ProcessHtmlForm(Sender: TObject);
    procedure SendHtmlHeader;
    procedure SendHtmlFooter;
    procedure UpdateActiveProvider;
    procedure CreateForm;

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

  ProviderID : String;
  TechID      : String;

implementation

{$R *.DFM}

procedure TForm1.ProcessHtmlForm(Sender: TObject);
begin
  with CGIEnvData1 do
    begin
      {required when this program runs under WebSite}
      webSiteINIFilename := paramstr(1);
      application.OnException := cgiErrorHandler;
      application.ProcessMessages;
    end;
  end;
end;
```

```

    {send header for cgi output to client}
    SendHtmlHeader;

    {Read in ProviderID and TechID from form literals}
    ProviderID := GetSmallField('ProviderID');
    TechID := GetSmallField('TechID');

    {Check Active Provider table and update time of last action}
    UpdateActiveProvider;

    {Send HTML form}
    CreateForm;
end;
end;

procedure TForm1.CreateForm;
begin
    with CGIEnvData1 do
    begin
        sendHdr('4','You have chosen to register an AMPL model schema with
            DecisionNet.');
```

send('</CENTER>');

send('You will be asked to:');

send('Provide the text of the model');

send('Provide an AMPL');

send('<a href="' + IniLink.StringEntry['OutputScriptPage'] +

'">output

'script for use with this model');

send('Specify a default solver for this model');

send('<form method=post action="' + IniLink.StringEntry['ExePath'] +

IniLink.StringEntry['NextExe'] + '">
');

SendHdr('4','1. Enter the model in the box below:
');

send('<textarea name="model" rows=20 cols=80>');

send('</textarea>

');

SendHdr('4','2. Enter the output script in the box below:
');

send('<textarea name="output" rows=20 cols=80>');

send('</textarea>

');

sendHdr('4','3. Choose a solver from the list below:');

Send('<SELECT NAME="SolverID" TYPE=TEXT>');

with QuerySolvers do

begin

open;

while not EOF do

begin

send('<OPTION VALUE="' + FieldByName('ProviderID').AsString +

'/' +

FieldByName('TechID').AsString + '">');

send(FieldByName('TechName').AsString + ' (' +

FieldByName('ProviderID').AsString + '-' +

FieldByName('TechID').AsString + ') ');

next;

end;

end;

send('</SELECT>

');

send('<input type=hidden name="ProviderID" value="' + ProviderID +

'">');

send('<input type=hidden name="TechID" value="' + TechID + '">');

send('<CENTER><input type=submit value="Submit

```

form"></CENTER></form>' );
end;
SendHTMLFooter;
end;

procedure TForm1.SendHtmlHeader;
begin
    with CGIEnvData1 do
    begin
        {HTML page header info}
        createStdout;
        sendPrologue;

        send( '<HTML><HEAD>' );
        sendTitle( 'DecisionNet AMPL Model Schema Registration' );
        send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
        sendHdr( '2','DecisionNet AMPL Model Schema Registration' );
        sendHR;

    end;
end;

procedure TForm1.SendHtmlFooter;
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}
        sendHR;
        send( 'This application was created by Michael Casey for Professor
            ' );
        send( 'Hemant Bhargava.<br>' );
        send( 'Generated on ' + webdate(now) );
        send( '</BODY></HTML>' );
        closeStdout;

        {quit application}
        closeApp(application);
    end;
end;

procedure TForm1.UpdateActiveProvider;
begin
    {Check Active Provider table and update time of last action}
    with ActProvTable, CGIEnvData1 do
    begin
        Open;
        If not FindKey([ProviderID]) then {Provider is not logged in}
        begin
            sendHdr('3',ProviderID + ' is not logged in. ');
            send('Please log in to DecisionNet and try again. ');
            SendHTMLFooter;
        end
        else {Update date and time of last action}
        begin
            Edit;
            FieldByName('LastActionTime').AsString := DateTimetoStr(now);
        end;
        Close;
    end;
end;

```

end;

end.

AMODREG2

```

unit Amodrg2;

{ This program is called by AMODREG to register an AMPL model. It adds
the model text, output script, and default AMPL solver to the Model
table. Control is then passed to TECHDONE to complete the Dnet portion
of the registration process.
  INPUTS: ProviderID, TechID, model, output, SolverID
  OUTPUTS: ProviderID, TechID
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, StdCtrls, Toolbar, ExtCtrls,
  UpdateOk, TpShell, IniLink, Ftp, starsock, DB, DBTables, Grids,
  DBGrids;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    TechTempTable: TTable;
    ActProvTable: TTable;
    TblModel: TTable;
    procedure ProcessHtmlForm(Sender: TObject);
    procedure SendHtmlHeader;
    procedure SendHtmlFooter;
    procedure UpdateActiveProvider;
    procedure CreateForm;
    procedure AddModel;

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

  ProviderID : String;
  TechID      : String;
  Model       : TStringList;
  Output      : TStringList;
  SolverID    : String;
  SolverProvID : String;
  SolverTechID : String;

implementation

{$R *.DFM}

procedure TForm1.ProcessHtmlForm(Sender: TObject);
begin
  with CGIEnvData1 do

```



```

begin
    {required when this program runs under WebSite}
    webSiteINIFilename := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;
    {send header for cgi output to client}
    SendHtmlHeader;

    {Read in ProviderID and TechID from form literals}
    ProviderID := GetSmallField('ProviderID');
    TechID := GetSmallField('TechID');

    {Check Active Provider table and update time of last action}
    UpdateActiveProvider;

    {Read in model text, output script, and default solver choice}
    Model := TStringList.create;
    Model.clear;
    GetTextArea( 'model' , Model );

    Output := TStringList.create;
    Output.clear;
    GetTextArea( 'output' , Output );

    SolverID := GetSmallField('SolverID');

    {Break SolverID into SolverProvID and SolverTechID}
    SolverProvID := Copy(SolverID, 0, Pos('/', SolverID)-1 );
    SolverTechID := Copy(SolverID, Pos('/', SolverID)+1 , 10);

    {Update database}
    AddModel;

    {Send HTML form}
    CreateForm;
end;
end;

procedure TForm1.AddModel;
begin
    with TblModel do
    begin
        open;
        AppendRecord(['', ProviderID, TechID, 'AMPL', Model, Output,
        SolverProvID,
        SolverTechID, 'Yes']);
        close;
    end;
end;

procedure TForm1.CreateForm;
begin
    with CGIEnvData1 do
    begin
        send('<CENTER>');
        sendHdr('4', 'Please press the button to complete the model schema '
+
        'registration process. ');
        send('<BR><BR><form method=post action="" +
        IniLink.StringEntry['ExePath'] +

```

```

        IniLink.StringEntry['NextExe'] + '>');
    send('<input type=hidden name="ProviderID" value="' + ProviderID +
        '>');
    send('<input type=hidden name="TechID" value="' + TechID + '>');
    send( '<CENTER><input type=submit value="Continue"></CENTER></form>'
        );
end;
SendHTMLFooter;
end;
procedure TForm1.SendHtmlHeader;
begin
    with CGIEnvData1 do
    begin
        {HTML page header info}
        createStdout;
        sendPrologue;

        send( '<HTML><HEAD>' );
        sendTitle( 'DecisionNet AMPL Model Schema Registration' );
        send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
        sendHdr( '2','DecisionNet AMPL Model Schema Registration' );
        sendHR;

    end;
end;

procedure TForm1.SendHtmlFooter;
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}
        sendHR;
        send( 'This application was created by Michael Casey for Professor
            ' );
        send( 'Hemant Bhargava.<br>' );
        send( 'Generated on ' + webdate(now) );
        send( '</BODY></HTML>' );
        closeStdout;

        {quit application}
        closeApp(application);
    end;
end;

procedure TForm1.UpdateActiveProvider;
begin
    {Check Active Provider table and update time of last action}
    with ActProvTable, CGIEnvData1 do
    begin
        Open;
        If not FindKey([ProviderID]) then {Provider is not logged in}
        begin
            sendHdr('3',ProviderID + ' is not logged in. ');
            send('Please log in to DecisionNet and try again. ');
            SendHTMLFooter;
        end
        else {Update date and time of last action}
        begin
            Edit;

```

```
        FieldByName('LastActionTime').AsString := DateTimetoStr(now);
    end;
    Close;
end;
end;
end.
```

UPDATE1

```
unit Update_1;

{ This program is called from DecisionNet when the user wishes to update
  (ie- modify) an AMPL object (either a model schema, dataset, or
  solver). It verifies that the chosen technology is still a registered
  AMPL dataset, model schema, or solver. It then gives the provider a form
  on which he can edit the data fields for the AMPL object.
  INPUTS: ProviderID, TechID
  OUTPUTS: ProviderID, TechID, SolverURL, Model, Output, SolverID,
  Dataset,
           ModelID.
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgldb, Cgi, Grids, DBGrids, DB, DBTables,
  StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    TechTable: TTable;
    TblModel: TTable;
    TblDataset: TTable;
    Query1: TQuery;
    QuerySolvers: TQuery;
    CGIDB1: TCGIDB;
    QueryModels: TQuery;
    procedure SendHtmlHeader;
    procedure Form1Create(Sender: TObject);
    procedure SendHtmlFooter;

  private
    { Private declarations }
  public
    { Public declarations }

  end;

var
  Form1: TForm1;

  ProviderID : string;
  TechID : string;
  ObjectType : string;
  ExcInd : string;
  TextLine : String;
  ModelNumber : String;
  DataSetNumber : String;
  SolverNumber : String;
```

```

implementation
{$R *.DFM}

procedure TForm1.Form1Create(Sender: TObject);

begin
  with CGIEnvData1 do
  begin
    {set up for CGI I/O}
    SendHtmlHeader;

    {retrieve input fields from HTML form}
    ProviderID := getSmallField( 'ProviderID' );
    TechID := getSmallField( 'TechID' );

    {Get object type from Technology table}
    with TechTable do
    begin
      Open;
      SetKey;
      FieldByName('ProviderID').AsString := ProviderID;
      FieldByName('TechID').AsString := TechID;
      If not GotoKey then {Technology not registered with DNet}
      begin
        sendHdr('3','This technology must first be registered with
          DecisionNet.');
```

send('Please go back to DecisionNet and try again.');

SendHTMLFooter;

end

```
      else {Get object type}
        ObjectType := FieldByName('tObjectType').AsString;
        ExcInd := FieldByName('ExcInd').AsString;
        Close;
      end;

      {Go to appropriate .exe for the type of object being executed}
      If not((ObjectType = 'Dataset') or (ObjectType = 'Model Schema') or
        (ObjectType = 'Solver')) then {ObjectType is not a valid AMPL
        object}
      begin
        sendHdr('3',ObjectType + ' is not a valid AMPL object.');
```

send('Allowed objects include: Model Schema, Dataset, and
 Solver.
');

send('Please go back to DecisionNet and try again.');

SendHTMLFooter;

end;

```
      If ExcInd <> 'AMPL' then {Specified object is not an AMPL
        technology}
      begin
        sendHdr('3',ObjectType + ' is not an AMPL technology.');
```

send('
');

send('Please go back to DecisionNet and try again.');

SendHTMLFooter;

end;

```
      {Open appropriate table for the type of object being updated}
      with Query1 do
      begin
```

```

        if ObjectType= 'Model Schema' then ObjectType := 'Model';
        close;
        SQL.clear;
        SQL.add( 'Select * from ' + ObjectType + ' where' );
        SQL.add( 'ProvidID="' + ProviderID + '" AND TechID="' + TechID +
        '";' );
        open;
        if EOF then {no table entry for this object. User probably tried
to
            change object type.}
        begin
            send( '<CENTER>' );
            sendHdr( '3','A DATABASE ERROR OCCURRED' );
            send( 'The most likely cause is that you changed the object type
            ');
            send( 'on the last form. You cannot change a registered AMPL
            object');
            send( 'from one type to another.<BR><BR></CENTER>' );
            send( '1. If you changed the object type by mistake, press your
            browser's' );
            send( 'BACK button until you are able to change the object type
            back ' );
            send( 'to the correct type.<BR>' );
            send( '2. If you did not change the object type, you should
            withdraw' );
            send( 'this technology and re-register it.<BR><BR><CENTER>' );
            send( '<form method=post action="' +
IniLink.StringEntry['ExePath'] +
                IniLink.StringEntry['MenuExe'] + '>');
            send( '<input type=hidden name="ProviderID" value="' +
ProviderID
                + '>');
            send( '<CENTER><input type=submit value="Return to Registered '
+
                'Provider Menu"></CENTER></form>' );
            SendHTMLFooter;
        end;

        {Send form with appropriate fields}
        send('<form method=post action="' +
IniLink.StringEntry['NextExe']);
        send('"><BR>');
        if ObjectType = 'Model' then
        begin
            {send form memo field containing model}
            SendHdr('3','Please update the model text as necessary.' );
            send( '<textarea name="Model" rows=20 cols=80>' );
            CGIDB1.sendMemo( FieldByName('Model') );
            send( '</textarea><BR><BR><HR>' );

            {send form memo field containing output script}
            SendHdr('3','Please update the execution script as necessary. ');
            send( '<textarea name="Output" rows=20 cols=80>' );
            CGIDB1.sendMemo( FieldByName('Output') );
            send( '</textarea><BR><BR>' );

            {send listbox containing choices for default solver}
            sendHdr('4', 'To change default AMPL solver, choose one from the
            list below:');
            Send('<SELECT NAME="SolverID" TYPE=TEXT>');

```

```

QuerySolvers.open;

{Send current default solver as first choice on list}
while not QuerySolvers.EOF do
begin
  if ((FieldByName('SolverProvID').AsString =
    QuerySolvers.FieldByName('ProviderID').AsString) AND
    (FieldByName('SolverTechID').AsString =
    QuerySolvers.FieldByName('TechID').AsString)) then begin
    send('<OPTION VALUE="' +
QuerySolvers.FieldByName('ProviderID').AsString + '/' +
    QuerySolvers.FieldByName('TechID').AsString + '>');
    send(QuerySolvers.FieldByName('TechName').AsString + ' (' +
    QuerySolvers.FieldByName('ProviderID').AsString + '-' +
    QuerySolvers.FieldByName('TechID').AsString + ') [Current
    default solver] ');
  end;
  QuerySolvers.Next;
end;

{List all registered AMPL solvers in order}
QuerySolvers.First;
while not QuerySolvers.EOF do
begin
  send('<OPTION VALUE="' +
QuerySolvers.FieldByName('ProviderID').AsString + '/' +
    QuerySolvers.FieldByName('TechID').AsString + '>');
  send(QuerySolvers.FieldByName('TechName').AsString + ' (' +
    QuerySolvers.FieldByName('ProviderID').AsString + '-' +
    QuerySolvers.FieldByName('TechID').AsString + ') ');
  QuerySolvers.next;
end;
  send('</select><BR><BR>');
end;
if ObjectType = 'Dataset' then
begin
  {send form memo field containing dataset}
  SendHdr('3', 'Please update the dataset text as necessary. ');
  send( '<textarea name="Dataset" rows=20 cols=80>' );
  CGIDB1.sendMemo( FieldByName('Dataset') );
  send( '</textarea><BR><BR><HR>' );

  {send listbox containing choices for default model}
  sendHdr('4', 'To change default AMPL model, choose one from the
  list below:');
  Send('<SELECT NAME="ModelID" TYPE=TEXT>');

  QueryModels.open;
  {Send current default Model as first choice on list}
  while not QuerySolvers.EOF do
  begin
    if ((FieldByName('ModelProvID').AsString =
      QueryModels.FieldByName('ProviderID').AsString) AND
      (FieldByName('ModelTechID').AsString =
      QueryModels.FieldByName('TechID').AsString)) then
    begin
      send('<OPTION VALUE="' +
QueryModels.FieldByName('ProviderID').AsString + '/' +
      QueryModels.FieldByName('TechID').AsString + '>');
      send(QueryModels.FieldByName('TechName').AsString + ' (' +

```

```

        QueryModels.FieldName('ProviderID').AsString + '-' +
        QueryModels.FieldName('TechID').AsString + ') [Current
        default model] ' );
    end;
    QueryModels.Next;
end;

{List all registered AMPL Models in order}
QueryModels.First;
while not QueryModels.EOF do
begin
    send('<OPTION VALUE="' +
QueryModels.FieldName('ProviderID').AsString + '/' +
        QueryModels.FieldName('TechID').AsString + '>');
    send(QueryModels.FieldName('TechName').AsString + ' (' +
        QueryModels.FieldName('ProviderID').AsString + '-' +
        QueryModels.FieldName('TechID').AsString + ') ' );
    QueryModels.next;
end;
    send('</select><BR><BR>');
end;
if ObjectType = 'Solver' then
begin
    sendHdr('4','Please update the URL for this solver as
necessary. ');
    send('<input type=text name="SolverURL" value="' +
        FieldByName('URL').AsString + '" size=60 maxlength=255>' );
    send('<P>');
end;
    send('<input type=hidden name="ProviderID" value="' + ProviderID +
        '>');
    send('<input type=hidden name="TechID" value="' + TechID + '>');
    send( '<input type=submit><input type=reset></form></CENTER>' );
end;

{Finish HTML form and close app}
SendHTMLFooter;
end;
end;

```

```

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin
    with CGIEnvData1 do
    begin
        {required when this program runs under WebSite}
        webSiteINIFileName := paramstr(1);
        application.onException := cgiErrorHandler;
        application.processMessages;

        {HTML page header info}
        createStdout;
        sendPrologue;

        send( '<HTML><HEAD>' );
        sendTitle( 'DecisionNet AMPL Object Update Form' );
        send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
        sendHdr( '2','Update DecisionNet AMPL Object' );
        sendHR;
    end;
end;

```



```

    end;
end;

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}
        send('<HR>');
        send( 'This application was created by Michael Casey for Professor
            ');
        send( 'Hemant Bhargava.<br>' );

        {timestamp}
        send( 'Generated on ' + webdate(now) );

        send( '</BODY></HTML>' );
        closeStdout;
        closeApp(application);
    end;
end;

end.

```

UPDATE2

```
unit Update_2;

{ This program is called from UPDATE1.EXE when the user has updated
  an AMPL object (either a model schema, dataset, or solver).
  It stores the updated fields in the appropriate ampldb table
  (SOLVER.DB, MODEL.DB, or DATASET.DB) and sends the user back to DNet's
  registered provider menu (MENU.EXE).
  INPUTS:  ProviderID, TechID, SolverURL, Model, Output, SolverID,
  Dataset,
           ModelID
  OUTPUTS: ProviderID
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, Grids, DBGrids, DB, DBTables,
  StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    TechTable: TTable;
    TblDataset: TTable;
    TblModel: TTable;
    QuerySolver: TQuery;
    procedure SendHtmlHeader;
    procedure Form1Create(Sender: TObject);
    procedure SendHtmlFooter;

  private
    { Private declarations }
  public
    { Public declarations }

  end;

var
  Form1: TForm1;

  ProviderID : string;
  TechID     : string;
  ObjectType  : string;
  ModelID    : string;
  SolverID   : String;
  SolverURL  : String;
  ModelProvID : String;
  ModelTechID : String;
  SolverProvID : String;
  SolverTechID : String;
  Model       : TStringList;
  Output      : TStringList;
  Dataset     : TStringList;
```

implementation

{SR *.DFM}

procedure TForm1.Form1Create(Sender: TObject);

begin

 with CGIEnvData1 do

 begin

 {set up for CGI I/O}

 SendHtmlHeader;

 {retrieve input fields from HTML form}

 ProviderID := getSmallField('ProviderID');

 TechID := getSmallField('TechID');

 SolverURL := getSmallField('SolverURL');

 SolverID := getSmallField('SolverID');

 ModelID := getSmallField('ModelID');

 Model := TStringList.create;

 Model.clear;

 GetTextArea('Model' , Model);

 Output := TStringList.create;

 Output.clear;

 GetTextArea('Output' , Output);

 Dataset := TStringList.create;

 Dataset.clear;

 GetTextArea('Dataset' , Dataset);

 {Get object type from Technology table}

 with TechTable do

 begin

 Open;

 SetKey;

 FieldByName('ProviderID').AsString := ProviderID;

 FieldByName('TechID').AsString := TechID;

 If GotoKey then

 ObjectType := FieldByName('tObjectType').AsString

 else

 begin

 send('No entry found in table TECHNOLO.DB!');

 SendHTMLFooter;

 end;

 Close;

 end;

 {Update appropriate table for the type of object being updated}

 if ObjectType = 'Model Schema' then

 with TblModel do

 begin

 {Break SolverID into SolverProvID and SolverTechID}

 SolverProvID := Copy(SolverID, 0, Pos('/', SolverID)-1);

 SolverTechID := Copy(SolverID, Pos('/', SolverID)+1 , 10);

 Open;

 FindKey([ProviderID, TechID]); {sets pointer to proper record}

 Delete; {Must delete then append because memo fields are involved}

```

        AppendRecord(['', ProviderID, TechID, 'AMPL', Model, Output,
        SolverProvID,
        SolverTechID, 'Yes']);
        Close;
    end;

    if ObjectType = 'Dataset' then
    with TblDataset do
    begin
        {Break ModelID into ModelProvID and ModelTechID}
        ModelProvID := Copy(ModelID, 0, Pos('/',ModelID)-1 );
        ModelTechID := Copy(ModelID, Pos('/',ModelID)+1 , 10);

        Open;
        FindKey( [ ProviderID, TechID ] ); {sets pointer to proper record}
        Delete; {Must delete then append because memo fields are involved}
        AppendRecord(['', ProviderID, TechID, 'AMPL', Dataset,
        ModelProvID,
        ModelTechID, 'Yes']);
        Close;
    end;

    if ObjectType = 'Solver' then
    with QuerySolver do
    begin
        Close;
        SQL.Clear;
        SQL.Add( 'Update Solver set URL = "' + SolverURL + '" );
        SQL.Add( 'where (ProvID="' + ProviderID + '" and TechID = "'
        + TechID + '");' );
        ExecSQL;
    end;

    send('<CENTER>');
    SendHdr('3','Your update has been accepted. ');
    send('Press the button to return to the DecisionNet Provider
    Menu. ');
    send('<BR><BR><form method=post action="' +
    IniLink.StringEntry['ExePath'] +
    IniLink.StringEntry['MenuExe'] + '>');
    send('<input type=hidden name="ProviderID" value="' + ProviderID +
    '>');
    send( '<input type=submit value="OK"></form></CENTER>' );

    {Finish HTML form and close app}
    SendHTMLFooter;
end;
end;

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin
    with CGIEnvData1 do
    begin
        {required when this program runs under WebSite}
        webSiteINIFileName := paramstr(1);
        application.onException := cgiErrorHandler;
        application.processMessages;
    end;
end;

```

```

    {HTML page header info}
    createStdout;
    sendPrologue;

    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Object Update Form' );
    send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
    sendHdr( '2','Update DecisionNet AMPL Object' );
    sendHR;
end;
end;

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}
        send('<HR>');
        send( 'This application was created by Michael Casey for Professor
            ' );
        send( 'Hemant Bhargava.<br>' );

        {timestamp}
        send( 'Generated on ' + webdate(now) );

        send( '</BODY></HTML>' );
        closeStdout;
        closeApp(application);
    end;
end;

end.

```

WITDRAW1

```
unit Witdrwl;
```

```
{ This program is called from DecisionNet when the user wishes to
  withdraw an AMPL object (either a model schema, dataset, or solver).
  For solvers, it lists all registered models using that solver as
  default. For models, it lists all datasets using that model as
  default. If any other AMPL objects use the selected object as default,
  this program requests verification via an HTML form that sends the
  user back here again. If no other object uses the selected object as a
  default, or if the user has already confirmed withdrawal, this program
  deletes the object from the appropriate AMPL table, removes all
  references to it from other AMPL objects, and calls DNet's WDTECHB.EXE
  to delete all DNet registration information on the dataset.
```

Note: the user will not be allowed to withdraw the AMPL default solver, which is specified in the file "dnetampl.ini". He will receive instead a message explaining that system administrator intervention is required to remove that solver.

```
INPUTS:  ProviderID, TechID, Confirmed
OUTPUTS: ProviderID, TechID, Confirmed
```

```
}
```

```
interface
```

```
uses
```

```
SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
Forms, Dialogs, Cgidb, Cgi, Grids, DBGrids, DB, DBTables,
StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;
```

```
type
```

```
TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  TpStatusBar1: TtpStatusBar;
  TpStatusPanell1: TtpStatusPanel;
  IniLink: TIniFileLink;
  TechTable: TTable;
  Query1: TQuery;
  IniLink2: TIniFileLink;
  procedure SendHtmlHeader;
  procedure Form1Create(Sender: TObject);
  procedure SendHtmlFooter;
  procedure DeleteModel;
  procedure DeleteSolver;
```

```
private
```

```
{ Private declarations }
```

```
public
```

```
{ Public declarations }
```

```
end;
```

```
var
```

```
Form1: TForm1;
```

```
ProviderID : string;
TechID      : string;
ObjectType  : string;
```

```

    TextLine      : String;
    ModelNumber   : String;
    DataSetNumber : String;
    SolverNumber  : String;
    Confirmed     : String;

implementation

{$R *.DFM}

procedure TForm1.Form1Create(Sender: TObject);

begin
    with CGIEnvData1 do
    begin
        {set up for CGI I/O}
        SendHtmlHeader;

        {retrieve input fields from HTML form}
        ProviderID := getSmallField( 'ProviderID' );
        TechID     := getSmallField( 'TechID' );
        Confirmed  := getSmallField( 'Confirmed' );

        {Get object type from Technology table}
        with TechTable do
        begin
            Open;
            SetKey;
            FieldByName('ProviderID').AsString := ProviderID;
            FieldByName('TechID').AsString := TechID;
            If not GotoKey then {Technology not registered with DNet}
            begin
                sendHdr('3','This technology must first be registered with
DecisionNet. ');
                send('Please go back to DecisionNet and try again. ');
                SendHTMLFooter;
            end
            else {Get object type}
                ObjectType := FieldByName('tObjectType').AsString;
            end;

            If (ObjectType <> 'Dataset') and (ObjectType <> 'Model Schema') and
                (ObjectType <> 'Solver') then {ObjectType is not a valid AMPL
object}
            begin
                sendHdr('3',ObjectType + ' is not a valid AMPL object. ');
                send('Allowed objects include: Model Schema, Dataset, and
Solver.<BR> ');
                send('Please go back to DecisionNet and try again. ');
                SendHTMLFooter;
            end;

            {Handle dataset withdrawal}
            if ObjectType = 'Dataset' then
            with Query1 do
            begin
                close;
                SQL.Clear;
                SQL.add( 'DELETE FROM DATASET WHERE (ProvID = "' + ProviderID +
                    '" and TechID = "' + TechID + '");' );
            end;
        end;
    end;
end;

```

```

        ExecSQL;
        send('<CENTER>');
        sendHdr('4','The dataset you selected has been deleted from the
AMPL      database.');
```

send(' Please press the button below to complete');

send(' the withdrawal process.');

send('<FORM method=post action="' +

```
IniLink.StringEntry['WdtechbExe'] + '>');
```

send('<input type=hidden name="ProviderID" value="' + ProviderID +

'>');

send('<input type=hidden name="TechID" value="' + TechID + '>');

send(' <input type=submit></form></CENTER>');

SendHTMLFooter;

end;

{Handle model withdrawal}

if ObjectType = 'Model Schema' then

if Confirmed = 'Yes' then {Delete model and remove references from

datasets}

begin

DeleteModel;

send('<CENTER>');

sendHdr('4','The model schema you selected has been deleted from

the AMPL database.');

send(' Please press the button below to complete');

send(' the withdrawal process.');

send('<FORM method=post action="' +

```
IniLink.StringEntry['WdtechbExe'] + '>');
```

send('<input type=hidden name="ProviderID" value="' + ProviderID

+ '>');

send('<input type=hidden name="TechID" value="' + TechID +

'>');

send(' <input type=submit VALUE="Continue"></form></CENTER>');

SendHTMLFooter;

end

else with Query1 do {List associated datasets and ask for

confirmation}

begin

close;

SQL.Clear;

SQL.add('SELECT * FROM DATASET WHERE (ModelProvID = "' +

ProviderID +

'" and ModelTechID = "' + TechID + '");');

open;

if not EOF then {at least one dataset uses this model as default}

begin

{send a table listing datasets which depend on this model}

send('<CENTER>');

SendHdr('4','Please consider the following...');

send('</CENTER>');

send('Some AMPL datasets registered with DecisionNet were

written');

send(' specifically for the model schema you have chosen to

withdraw.');

send(' These datasets use this model schema as a default option.

');

send('These datasets are listed in the table below. If you are

the ');

send('provider of any of these datasets, we highly encourage you

to ');


```

        send('either withdraw them or update them with a different model
            schema.');
```

Provider	Tech ID	Dataset
<pre> while not EOF do {once for each dataset in list} begin TechTable.FindKey([FieldByName('ProvID').AsString, FieldByName('TechID').AsString]); {synchs TECHNOLO.DB w/ query} send('<TR><TD>' + FieldByName('ProvID').AsString + '</TD><TD>' + FieldByName('TechID').AsString + '</TD><TD>'); send(TechTable.FieldByName('TechName').AsString + '</TD><TR>'); next; end; send('</TABLE><P>'); send('<FORM method=post action="' + IniLink.StringEntry['ThisExe'] + '">'); send('<input type=hidden name="ProviderID" value="' + ProviderID + '">'); send('<input type=hidden name="TechID" value="' + TechID + '">'); send('<input type=hidden name="Confirmed" value="Yes">'); send('<input type=submit value="Withdraw Model Schema"></form>'); send('<FORM method=post action="' + IniLink.StringEntry['MenuExe'] + '">'); send('<input type=hidden name="ProviderID" value="' + ProviderID + '">'); send('<input type=submit value="Cancel, Return to Provider Menu"></form></CENTER>'); SendHTMLFooter; end else begin DeleteModel; send('<CENTER>'); sendHdr('4','The model schema you selected has been deleted from the AMPL database.');</pre>		

```

            send(' Please press the button below to complete');
            send(' the withdrawal process.');
```

<pre> send('<FORM method=post action="' + IniLink.StringEntry['WdtechbExe'] + '">'); send('<input type=hidden name="ProviderID" value="' + ProviderID + '">'); send('<input type=hidden name="TechID" value="' + TechID + '">'); send('<input type=submit VALUE="Continue"></form></CENTER>'); SendHTMLFooter; end; end;</pre>
--

```

        {Handle solver withdrawal for default solver}
        if ((ObjectType = 'Solver') and
            (IniLink2.StringEntry['DefaultSolverProvID']
                = ProviderID) and (IniLink2.StringEntry['DefaultSolverTechID'] =

```

```

TechID))
then
begin
  send('<CENTER>');
  sendHdr('4','Withdrawal of this solver requires System
Administrator approval.');
```

send('</CENTER>');
 send('The DecisionNet AMPL system uses this solver as a default
 solver');
 send('in situations where no other solver is specified by a
 consumer');
 send('or provider. If you wish to remove this solver from the
 system,');
 send('contact the DecisionNet staff, so they may switch the
 system');
 send('default to another AMPL solver. Then you will be allowed
 to');
 send('withdraw this solver via the method you just tried. Please
 keep the');
 send('solver operational until the withdrawal process is
 complete.');

```

  send('<FORM method=post action="' +
IniLink.StringEntry['MenuExe']          + '>');
  send('<input type=hidden name="ProviderID" value="' + ProviderID
+
  '>');
  send('<CENTER>');
  send('<input type=submit value="Back to Provider
Menu"></form></CENTER>');
  SendHTMLFooter;
end;

(Handle solver withdrawal for any solver except default solver)
if ObjectType = 'Solver' then
  if Confirmed = 'Yes' then {Delete solver and remove references
from
  model schemas}
  begin
    DeleteSolver;
    send('<CENTER>');
    sendHdr('4','The solver you selected has been deleted from the
    AMPL database.');
```

send(' Please press the button below to complete');
 send(' the withdrawal process. ');
 send('<FORM method=post action="' +
IniLink.StringEntry['WdtechbExe'] + '>');
 send('<input type=hidden name="ProviderID" value="' +
ProviderID
 + '>');
 send('<input type=hidden name="TechID" value="' + TechID +
 '>');
 send(' <input type=submit VALUE="Continue"></form></CENTER>'

);

```

  SendHTMLFooter;
  end
  else with Query1 do {List associated model schemas and ask for
  confirmation}
  begin
    close;
    SQL.Clear;
    SQL.add( 'SELECT * FROM MODEL WHERE (SolverProvID = "' +
ProviderID +
    '" and SolverTechID = "' + TechID + '");' );

```

```

open;
if not EOF then {at least one model schema uses this solver as
default}
begin
  {send a table listing model schemas which depend on this
solver}
  send('<CENTER>');
  SendHdr('4','Please consider the following...');
  send('Some AMPL model schemas registered with DecisionNet
were          registered');
  send(' using this solver as a default option. ');
  send('These model schemas are listed in the table below. If
you are the ');
  send('provider of any of these model schemas, we highly
encourage you to ');
  send('update them with a different default solver.
Otherwise,          the overall ');
  send('default solver for the DecisionNet AMPL system will be
substituted. ');
  send('<BR>');
  send('<TABLE BORDER=6 CELLPADDING=6>');
  send('<TR><TH>Provider</TH><th>Tech ID</th><th>Model Schema
Name</th></TR>');
  while not EOF do {once for each solver in list}
  begin
    TechTable.FindKey([ FieldByName('ProvID').AsString,
      FieldByName('TechID').AsString ]);
    send('<TR><TD>'+FieldByName('ProvID').AsString+'</TD><TD>'+
      FieldByName('TechID').AsString+'</TD><TD>'+
      TechTable.FieldByName('TechName').AsString + '</TD><TR>'
      );
    next;
  end;
  send(' </TABLE><P> ');
  send('<FORM method=post action="" +
IniLink.StringEntry['ThisExe'] + ">');
  send('<input type=hidden name="ProviderID" value="" +
ProviderID + ">');
  send('<input type=hidden name="TechID" value="" + TechID +
">');
  send('<input type=hidden name="Confirmed" value="Yes">');
  send(' <input type=submit value="Withdraw Model
Schema"></form> ');
  send('<FORM method=post action="" +
IniLink.StringEntry['MenuExe'] + ">');
  send('<input type=hidden name="ProviderID" value="" +
ProviderID + ">');
  send(' <input type=submit value="Cancel, Return to Provider
Menu"></form></CENTER> ');
  SendHTMLFooter;
end
else {no registered models list this solver as default}
begin
  DeleteSolver;
  send('<CENTER>');
  sendHdr('4','The solver you selected has been deleted from
the          AMPL database. ');
  send(' Please press the button below to complete ');
  send(' the withdrawal process. ');

```

```

        send('<FORM method=post action="' +
IniLink.StringEntry['WdtechbExe'] + '>');
        send('<input type=hidden name="ProviderID" value="' +
ProviderID + '>');
        send('<input type=hidden name="TechID" value="' + TechID +
'>');
        send(' <input type=submit></form></CENTER>' );
        SendHTMLFooter;
    end;
end; {if solver statement}
end;
end;

```

```

procedure TForm1.DeleteModel;
begin
    {This procedure removes the specified model schema, as well as all
    references
    to it in registered datasets.}
    with Query1 do
    begin
        close;
        SQL.clear;
        SQL.add( 'delete from model where (ProvID="' + ProviderID +
' " and TechID="' + TechID + '");' );
        ExecSQL;
        close;
        SQL.clear;
        SQL.add( 'update dataset set ModelProvID="", ModelTechID="" where
(ModelProvID=" '
+ ProviderID + ' " and ModelTechID=" ' + TechID + '");' );
        ExecSQL;
    end;
end;

```

```

procedure TForm1.DeleteSolver;
begin
    {This procedure removes the specified solver, as well as all
    references
    to it in registered model schemas.}
    with Query1 do
    begin
        close;
        SQL.clear;
        SQL.add( 'delete from solver where (ProvID="' + ProviderID +
' " and TechID=" ' + TechID + '");' );
        ExecSQL;
        close;
        SQL.clear;
        SQL.add('update model set SolverProvID="'+
IniLink2.StringEntry['DefaultSolverProvID']+
' ", SolverTechID="'+IniLink2.StringEntry['DefaultSolverTechID']+
' " where (SolverProvID="'+ ProviderID + ' " and SolverTechID=" ' +
TechID + '");' );
        ExecSQL;
    end;
end;

```

```

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin

```

```

with CGIEnvData1 do
begin
    {required when this program runs under WebSite}
    webSiteINIFileName := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {HTML page header info}
    createStdout;
    sendPrologue;

    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Object Withdrawal Form' );
    send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
    sendHdr( '2', 'Withdraw DecisionNet AMPL Object' );
    sendHR;
end;
end;

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}
        send('<HR>');
        send( 'This application was created by Michael Casey for Professor
            ' );
        send( 'Hemant Bhargava.<br>' );

        {timestamp}
        send( 'Generated on ' + webdate(now) );

        send( '</BODY></HTML>' );
        closeStdout;
        closeApp(application);
    end;
end;
end.

```

AMPLEXEC

```
unit Amplexel;

{ This program is called from DecisionNet when the user wishes to
execute
  an AMPL object (either a model schema, dataset, or solver). It
verifies
  that the UserID is an active consumer or provider, and that the chosen
technology is a registered AMPL dataset, model schema, or solver. It
then
  passes the ProviderID and TechID of the chosen object to the
appropriate
  program for that object type: AMPLEXE2 for model schemas, AMPLEXE3 for
datasets, and AMPLEXE4 for solvers.
INPUTS: UserID, ProviderID, TechID
OUTPUTS: ProviderID, TechID
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgldb, Cgi, Grids, DBGrids, DB, DBTables, Ftp,
  starsock,
  StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    ActProvTable: TTable;
    ActConstTable: TTable;
    TechTable: TTable;
    procedure SendHtmlHeader;
    procedure Form1Create(Sender: TObject);
    procedure SendHtmlFooter;
    function UpdateActiveProvider: Boolean;
    function UpdateActiveConsumer: Boolean;
    procedure VerifyUser;

  private
    { Private declarations }
  public
    { Public declarations }

  end;

var
  Form1: TForm1;

  UserID : string;
  ProviderID : string;
  TechID : string;
  ObjectType : string;
  ExcInd : string;
  TextLine : String;
  ModelNumber : String;
```

```

DataSetNumber : String;
SolverNumber : String;

```

implementation

```
{SR *.DFM}
```

```
procedure TForm1.Form1Create(Sender: TObject);
```

```
begin
```

```
  with CGIEnvData1 do
```

```
    begin
```

```
      {set up for CGI I/O}
```

```
      SendHtmlHeader;
```

```
      {retrieve input fields from HTML form}
```

```
      UserID := getSmallField( 'UserID' );
```

```
      ProviderID := getSmallField( 'ProviderID' );
```

```
      TechID := getSmallField( 'TechID' );
```

```
      {Verify UserID as an active consumer or provider}
```

```
      VerifyUser;
```

```
      {Get object type from Technology table}
```

```
      with TechTable do
```

```
        begin
```

```
          Open;
```

```
          SetKey;
```

```
          FieldByName('ProviderID').AsString := ProviderID;
```

```
          FieldByName('TechID').AsString := TechID;
```

```
          If not GotoKey then {Technology not registered with DNet}
```

```
            begin
```

```
              sendHdr('3','This technology must first be registered with  
DecisionNet.');
```

```
              send('Please go back to DecisionNet and try again.');
```

```
              SendHTMLFooter;
```

```
            end
```

```
          else {Get object type}
```

```
            ObjectType := FieldByName('tObjectType').AsString;
```

```
            ExcInd := FieldByName('ExcInd').AsString;
```

```
            Close;
```

```
          end;
```

```
          {Go to appropriate .exe for the type of object being executed}
```

```
          If (ObjectType <> 'Dataset') and (ObjectType <> 'Model Schema') and
```

```
            (ObjectType <> 'Solver') then {ObjectType is not a valid AMPL  
object}
```

```
            begin
```

```
              sendHdr('3',ObjectType + ' is not a valid AMPL object.');
```

```
              send('Allowed objects include: Model Schema, Dataset, and
```

```
Solver.<BR>');
```

```
              send('Please go back to DecisionNet and try again.');
```

```
              SendHTMLFooter;
```

```
            end;
```

```
          If ExcInd <> 'AMPL' then {Specified object is not an AMPL  
technology}
```

```

begin
    sendHdr('3',ObjectType + ' is not an AMPL technology.');
```

send('
');

send('Please go back to DecisionNet and try again.');

SendHTMLFooter;

end;

{Go to appropriate .exe to continue}

send('Please press the button to begin setup for the AMPL ');

send(AnsiLowerCase(ObjectType) + ' you specified.');

if ObjectType = 'Dataset' then

send('<form method=post action="' + IniLink.StringEntry['ExePath']

+ IniLink.StringEntry['DatasetExe']);

if ObjectType = 'Model Schema' then

send('<form method=post action="' + IniLink.StringEntry['ExePath']

+ IniLink.StringEntry['ModelExe']);

if ObjectType = 'Solver' then

send('<form method=post action="' + IniLink.StringEntry['ExePath']

+ IniLink.StringEntry['SolverExe']);

send('>
');

send('<input type=hidden name="ProviderID" value="' + ProviderID +

'">');

send('<input type=hidden name="TechID" value="' + TechID + '">');

send('<input type=submit value="OK"></form></CENTER>');

{Finish HTML form and close app}

SendHTMLFooter;

end;

end;

```

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin
    with CGIEnvData1 do
    begin
        {required when this program runs under WebSite}
        webSiteINIFileName := paramstr(1);
        application.onException := cgiErrorHandler;
        application.processMessages;

        {HTML page header info}
        createStdout;
        sendPrologue;

        send( '<HTML><HEAD>' );
        sendTitle( 'DecisionNet AMPL Execution' );
        send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
        sendHdr( '2','DecisionNet AMPL Execution' );
        sendHR;
    end;
end;
```

```

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin
```



```

with CGIEnvData1 do
begin
  {HTML page footer info}
  send('<HR>');
  send( 'This application was created by Michael Casey for Professor
    ');
  send( 'Hemant Bhargava.<br>' );

  {timestamp}
  send( 'Generated on ' + webdate(now) );

  send( '</BODY></HTML>' );
  closeStdout;
  closeApp(application);
end;
end;

```

```

procedure TForm1.VerifyUser;
{Checks UserID first in ACT_CONS then in ACT_PROV}
{and gives HTML error message if not found}
begin
  if not UpdateActiveConsumer then
    if not UpdateActiveProvider then
      {UserID not found as consumer or provider}
      with CGIEnvData1 do
      begin
        sendHdr('3','User ' + UserID + ' is not logged in.');
```

```

        send('Please log in to DecisionNet and try again.');
```

```

        SendHTMLFooter;
      end;
    end;
  end;
end;

function TForm1.UpdateActiveProvider: Boolean;
begin
  {Check Active Provider table and update time of last action}
  with ActProvTable, CGIEnvData1 do
  begin
    Open;
    If not FindKey([ProviderID]) then {Provider is not logged in}
      UpdateActiveProvider := False
    else {Update date and time of last action}
      begin
        UpdateActiveProvider := True;
        Edit;
        FieldByName('LastActionTime').AsString := DateTimetoStr(now);
      end;
    Close;
  end;
end;

function TForm1.UpdateActiveConsumer: Boolean;
begin
  {Check Active Consumer table and update time of last action}
  with ActConsTable, CGIEnvData1 do
  begin
    Open;
    If not FindKey([UserID]) then {Consumer is not logged in}
      UpdateActiveConsumer := False
    else {Update date and time of last action}

```

```
begin
  UpdateActiveConsumer := True;
  Edit;
  FieldByName('LastActionTime').AsString := DateTimeToStr(now);
end;
Close;
end;
end;
end.
```

AMPLEXE2

unit Ampexe2;

{ This program is called by AMPLEXEC when a consumer wishes to execute an AMPL model schema. The user is prompted to either select a registered dataset (from list) or enter his own dataset. The user's response is processed by the next program called, AMPLEXE5.

INPUTS: ProviderID, TechID

OUTPUTS: Method, DatasetID, Dataset, ModelProvID, ModelTechID,
SolverURL
}

interface

uses

SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
Forms, Dialogs, Cgldb, Cgi, Grids, DBGrids, DB, DBTables, Ftp,
starsock,
StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;

type

TForm1 = class(TForm)
 CGIEnvData1: TCGIEnvData;
 TpStatusBar1: TtpStatusBar;
 TpStatusPanell: TtpStatusPanel;
 IniLink: TIniFileLink;
 TblModel: TTable;
 QueryDatasets: TQuery;
 procedure SendHtmlHeader;
 procedure Form1Create(Sender: TObject);
 procedure SendHtmlFooter;

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

ProviderID : string;

TechID : string;

implementation

{\$R *.DFM}

procedure TForm1.Form1Create(Sender: TObject);

begin

with CGIEnvData1 do

begin

{set up for CGI I/O}

SendHtmlHeader;

{retrieve input fields from HTML form}

ProviderID := getSmallField('ProviderID');

```

TechID := getSmallField( 'TechID' );

{Send first option for specifying a dataset}
Send('<FORM method=post action="' + IniLink.StringEntry['ExePath'] +
  IniLink.StringEntry['NextExe'] + '>');
Send('<INPUT TYPE=RADIO NAME="Method" VALUE="1">');
Send('<B>Method 1: Select a registered dataset to use with this
model      schema.</B><BR>');
Send('Note: Not all datasets are compatible with this model schema.
  Use only ' +
  'datasets that you know to be compatible.<BR>');
Send('<CENTER><SELECT NAME="DatasetID" TYPE=TEXT >');
with QueryDatasets do
begin
  Close;
  Open;
  while not EOF do
  begin
    send('<OPTION VALUE="' +
      FieldByName('ProviderID').AsString + '/' +
      FieldByName('TechID').AsString + '>');
    send(FieldByName('TechName').AsString + ' (' +
      FieldByName('ProviderID').AsString + '-' +
      FieldByName('TechID').AsString + ')');
    next;
  end;
end;
send('</SELECT></CENTER><BR>');
SendHR;

{Send second option for specifying a dataset}
Send('<BR><INPUT TYPE=RADIO NAME="Method" VALUE="2">');
Send('<B>Method 2: Type or paste in a dataset to use with this model
  schema.</B><BR>');
Send('<BR><CENTER>');
SendHdr('4', 'Enter the dataset in the box below:<BR>');
send( '<textarea name="Dataset" rows=20 cols=80>' );
send( '</textarea><BR><BR><HR></CENTER>' );

{Finish HTML form}
send( '<INPUT TYPE=HIDDEN NAME="ModelProvID"
VALUE="' + ProviderID + '>' );
send( '<INPUT TYPE=HIDDEN NAME="ModelTechID" VALUE="' + TechID + '>' );
{pass on the solver URL, if a solver was chosen first}
if getSmallField('SolverURL') <> cginotfound then
  send( '<INPUT TYPE=HIDDEN NAME="SolverURL" VALUE="' +
    getSmallField('SolverURL') + '>' );
send( '<CENTER><INPUT TYPE=SUBMIT VALUE="Submit">' );
send( '<INPUT TYPE=RESET VALUE="Reset Form">' );
send( '</CENTER></FORM>' );

  SendHtmlFooter;
end;
end;

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin
  with CGIEnvData1 do
  begin

```

```

{required when this program runs under WebSite}
webSiteINIFileName := paramstr(1);
application.onException := cgiErrorHandler;
application.processMessages;

{HTML page header info}
createStdout;
sendPrologue;

send( '<HTML><HEAD>' );
sendTitle( 'DecisionNet AMPL Execution' );
send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
sendHdr( '2','DecisionNet AMPL Model Execution' );
sendHdr('3','Choose one of the methods below to specify a dataset to
    be' +
    ' used with this model schema.</CENTER>');
sendHR;
end;
end;

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}

        send( 'This application was created by Michael Casey for Professor
            ');
        send( 'Hemant Bhargava.<br>' );

        {timestamp}
        send( 'Generated on ' + webdate(now) );

        send( '</BODY></HTML>' );
        closeStdout;
        closeApp(application);
    end;
end;

end.

```

AMPLEXE3

```
unit Ampexe3;

{ This program is called by AMPLEXEC when a consumer wishes to execute
an AMPL dataset. The user is prompted to either select the default
model, select another registered model (from list) or enter his own
model. The user's response is processed by the next program called,
AMPLEXE6.
  INPUTS:  ProviderID, TechID, SolverURL
  OUTPUTS: DefaultProvID, DefaultTechID, Method, ModelID, model, output,
           SolverID, SolverURL, DSProvID, DSTechID
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgldb, Cgi, Grids, DBGrids, DB, DBTables, Ftp,
  starsock,
  StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanell: TtpStatusPanel;
    IniLink: TIniFileLink;
    QueryModels: TQuery;
    QuerySolvers: TQuery;
    QueryDataset: TQuery;
    TblDefaultModel1: TTable;
    CGIDB1: TCGIDB;
    TblDefaultModel2: TTable;
    procedure SendHtmlHeader;
    procedure Form1Create(Sender: TObject);
    procedure SendHtmlFooter;

  private
    { Private declarations }
  public
    { Public declarations }

  end;

var
  Form1: TForm1;

  ProviderID : string;
  TechID      : string;
  SolverURL   : string;

implementation

{$R *.DFM}

procedure TForm1.Form1Create(Sender: TObject);

begin
  with CGIEnvData1 do
```

```

begin
  {set up for CGI I/O}
  SendHtmlHeader;

  {retrieve input fields from HTML form}
  ProviderID := getSmallField( 'ProviderID' );
  TechID := getSmallField( 'TechID' );
  SolverURL := getSmallField('SolverURL');

  {Query to get fields of dataset record}
  with QueryDataSet do
  begin
    Close;
    SQL.Clear;
    SQL.Add('SELECT * FROM DATASET WHERE');
    SQL.Add('(ProvID="' + ProviderID + '" AND TechID="' + TechID +
'");' );
    Open;
  end;

  {Query to get all AMPL solvers}
  QuerySolvers.Close;
  QuerySolvers.Open;

  {Send first option for specifying a model}

  Send('<FORM method=post action="' + IniLink.StringEntry['ExePath'] +
IniLink.StringEntry['NextExe'] + '>');
  Send('<INPUT TYPE=HIDDEN NAME="DefaultProvID" VALUE="' +
QueryDataset.FieldName('ModelProvID').AsString + '>' );
  Send('<INPUT TYPE=HIDDEN NAME="DefaultTechID" VALUE="' +
QueryDataset.FieldName('ModelTechID').AsString + '>' );
  Send('<INPUT TYPE=RADIO NAME="Method" VALUE="1">');
  Send('<B>Method 1: Use the default model for this dataset, as
described below.</B>');

  {Get info on default model for this dataset}
  with TblDefaultModel1, TblDefaultModel2 do
  begin
    TblDefaultModel1.Open;
    TblDefaultModel1.SetKey;
    TblDefaultModel1.FieldName('ProviderID').AsString :=
QueryDataset.FieldName('ModelProvID').AsString;
    TblDefaultModel1.FieldName('TechID').AsString :=
QueryDataset.FieldName('ModelTechID').AsString;
    if not TblDefaultModel1.GotoKey then {send msg saying default
model invalid}
    begin
      Send('<BR>The default model for this dataset was not found.
Choose another option.');
```

```

send('<p>');

send('<center>');

send('<TABLE BORDER=6 CELLPADDING=6>');

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Technology Name:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(TblDefaultModell.FieldByName('TechName').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Provider ID:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(TblDefaultModell.FieldByName('ProviderID').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Technology ID:</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(TblDefaultModell.FieldByName('TechID').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Object Type</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(TblDefaultModell.FieldByName('tObjectType').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Problem Area</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(TblDefaultModell.FieldByName('tProblemArea').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Functional Area</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');

send(TblDefaultModell.FieldByName('tFunctionalArea').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Solution Method</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');

send(TblDefaultModell.FieldByName('tSolutionMethod').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Industry Type</B>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send(TblDefaultModell.FieldByName('tIndType').AsString);

send('<TR ALIGN="CENTER" VALIGN=MIDDLE>');
send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
send('<B>Organization Type</B>');

```



```

        send('<TD ALIGN="CENTER" VALIGN=MIDDLE>');
        send(TblDefaultModel1.FieldByName('tOrgType').AsString);
        send('</TABLE>');

        send('<P>');
        send('<B>Purpose:</B> ');
        CGIDB1.SendMemo(TblDefaultModel2.FieldByName('Purpose'));
        send('<P>');

        send('<P>');
        send('<B>Comments:</B> ');
        CGIDB1.SendMemo(TblDefaultModel2.FieldByName('Comments'));
        send('</center>');
    end;
end;

{Send second option for specifying a model}
SendHR;
Send('<INPUT TYPE=RADIO NAME="Method" VALUE="2">');
Send('<B>Method 2: Select another registered model to use with this
dataset.</B><BR>');
Send('Note: Not all models are compatible with this dataset. Use
only ' +
    'models that you know to be compatible.<BR>');
Send('<CENTER><SELECT NAME="ModelID" TYPE=TEXT >');
with QueryModels do
begin
    Close;
    Open;
    while not EOF do
    begin
        send('<OPTION VALUE="' +
            FieldByName('ProviderID').AsString + '/' +
            FieldByName('TechID').AsString + '>');
        send(FieldByName('TechName').AsString + ' (' +
            FieldByName('ProviderID').AsString + '-' +
            FieldByName('TechID').AsString + ')');
    next;
    end;
end;
send('</SELECT></CENTER>');
SendHR;

{Send third option for specifying a model}
Send('<INPUT TYPE=RADIO NAME="Method" VALUE="3">');
Send('<B>Method 3: Type or paste in a model to use with this
dataset.</B><BR>');
Send('Note: To use this option, you must also specify an AMPL output
+ 'script');
if SolverURL = cginotfound then {user didn't already select a
solver}
    send(' and select a solver');
send('.<BR><BR><CENTER>');
SendHdr('4','Enter the model in the box below:<BR>');
send(' <textarea name="model" rows=20 cols=80>');
send(' </textarea><BR><BR>');
SendHdr('4','Enter the output script in the box below:<BR>');
send(' <textarea name="output" rows=20 cols=80>');

```

```

send( '</textarea><BR><BR>' );
if SolverURL = cginotfound then {prompt user to select a solver}
begin
  sendHdr('4', 'Choose a solver from the list below:');
  Send('<SELECT NAME="SolverID" TYPE=TEXT >');
  with QuerySolvers do
  begin
    First;
    while not EOF do
    begin
      send('<OPTION VALUE="' + FieldByName('ProviderID').AsString +
        '/' +
        FieldByName('TechID').AsString + '>');
      send(FieldByName('TechName').AsString + ' (' +
        FieldByName('ProviderID').AsString + '-' +
        FieldByName('TechID').AsString + ')');
      next;
    end;
  end;
  send('</SELECT>');
end;

{Finish HTML form}
sendHR;
send( '<INPUT TYPE=HIDDEN NAME="DSProvID" VALUE="' + ProviderID + '>'
);
send( '<INPUT TYPE=HIDDEN NAME="DSTechID" VALUE="' + TechID + '>' );
{pass on the solver URL, if a solver was chosen first}
if SolverURL <> cginotfound then
  send( '<INPUT TYPE=HIDDEN NAME="SolverURL" VALUE="' + SolverURL +
    '>' );
send( '<INPUT TYPE=SUBMIT VALUE="Submit">' );
send( '<INPUT TYPE=RESET VALUE="Reset Form">' );
send( '</CENTER></FORM>' );

  SendHtmlFooter;
end;
end;

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin
  with CGIEnvData1 do
  begin
    {required when this program runs under WebSite}
    webSiteINIFileName := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {HTML page header info}
    createStdout;
    sendPrologue;

    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Execution' );
    send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
    sendHdr( '2', 'DecisionNet AMPL Dataset Execution' );
    sendHdr( '3', 'Choose one of the three methods below to specify a
model      to be' +
      ' used with the dataset you selected.</CENTER>' );

```

```

        sendHR;
    end;
end;

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}

        send( 'This application was created by Michael Casey for Professor
            ');
        send( 'Hemant Bhargava.<br>' );

        {timestamp}
        send( 'Generated on ' + webdate(now) );

        send( '</BODY></HTML>' );
        closeStdout;
        closeApp(application);
    end;
end;

end.

```

AMPLEXE4

```
unit Ampexe4;

{ This program is called by AMPLEEXEC when a consumer wishes to use an
  AMPL solver. The user is prompted to either select a registered
  model schema (from list), select a registered dataset (from list) or
  enter his own model schema and dataset. The user's response is
  processed by the next program called, AMPLEXE7.
  INPUTS: ProviderID, TechID
  OUTPUTS: Method, SolverURL, ModelID, DatasetID, model, dataset, output
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgldb, Cgi, Grids, DBGrids, DB, DBTables, Ftp,
  starsock,
  StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    QueryDataset: TQuery;
    QueryModel: TQuery;
    QuerySolver: TQuery;
    procedure SendHtmlHeader;
    procedure Form1Create(Sender: TObject);
    procedure SendHtmlFooter;

  private
    { Private declarations }
  public
    { Public declarations }

  end;

var
  Form1: TForm1;

  ProviderID : string;
  TechID : string;

implementation

{$R *.DFM}

procedure TForm1.Form1Create(Sender: TObject);
begin
  with CGIEnvData1 do
  begin
    {set up for CGI I/O}
    SendHtmlHeader;

    {retrieive input fields from HTML form}
  end;
end;
```

```

ProviderID := getSmallField( 'ProviderID' );
TechID := getSmallField( 'TechID' );

{Query to get Solver URL}
with QuerySolver do
begin
  close;
  SQL.Clear;
  SQL.Add('SELECT URL FROM SOLVER WHERE');
  SQL.Add('((ProvID="' + ProviderID + '" AND');
  SQL.Add('(TechID="' + TechID + '"))');
  open;
end;

{Send first option [for specifying a model]}
Send('<FORM method=post action="' + IniLink.StringEntry['ExePath'] +
  IniLink.StringEntry['NextExe'] + '">');
Send('<INPUT TYPE=RADIO NAME="Method" VALUE="1">');
Send('<B>Method 1: Select a registered model to use with this
solver.</B><BR>');
Send('<INPUT TYPE=HIDDEN NAME="SolverURL" VALUE="' +
  QuerySolver.FieldName('URL').AsString + '">');
Send('<CENTER><SELECT NAME="ModelID" TYPE=TEXT>');
with QueryModel do
begin
  Close;
  Open;
  while not EOF do
  begin
    send('<OPTION VALUE="' +
      FieldByName('ProviderID').AsString + '/' +
      FieldByName('TechID').AsString + '">');
    send(FieldByName('TechName').AsString + ' (' +
      FieldByName('ProviderID').AsString + '-' +
      FieldByName('TechID').AsString + ') ');
    next;
  end;
end;
send('</SELECT></CENTER>');
SendHR;

{Send second option [for specifying a dataset]}
Send('<INPUT TYPE=RADIO NAME="Method" VALUE="2">');
Send('<B>Method 2: Select a registered dataset to use with this
solver.</B><BR>');
Send('<CENTER><SELECT NAME="DatasetID" TYPE=TEXT>');
with QueryDataset do
begin
  Close;
  Open;
  while not EOF do
  begin
    send('<OPTION VALUE="' +
      FieldByName('ProviderID').AsString + '/' +
      FieldByName('TechID').AsString + '">');
    send(FieldByName('TechName').AsString + ' (' +
      FieldByName('ProviderID').AsString + '-' +
      FieldByName('TechID').AsString + ') ');
    next;
  end;
end;

```

```

end;
send('</SELECT>');
SendHR;

{Send third option for specifying a model and dataset}
Send('<INPUT TYPE=RADIO NAME="Method" VALUE="3">');
Send('<B>Method 3: Type or paste in a model and dataset to use with
      this solver.</B><BR>');
Send('Note: To use this option, you must also specify an AMPL output
,
      + 'script.<BR><BR><CENTER>');
SendHdr('4','Enter the model in the box below:<BR>');
send( '<textarea name="model" rows=20 cols=80>' );
send( '</textarea><BR><BR>' );
SendHdr('4','Enter the dataset in the box below:<BR>');
send( '<textarea name="dataset" rows=20 cols=80>' );
send( '</textarea><BR><BR>' );
SendHdr('4','Enter the output script in the box below:<BR>');
send( '<textarea name="output" rows=20 cols=80>' );
send( '</textarea><BR><BR>' );

{Finish HTML form}
send( '<INPUT TYPE=SUBMIT VALUE="Submit">' );
send( '<INPUT TYPE=RESET VALUE="Reset Form">' );
send( '</CENTER></FORM><HR>' );

SendHtmlFooter;
end;
end;

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin
  with CGIEnvData1 do
    begin
      {required when this program runs under WebSite}
      webSiteINIFileName := paramstr(1);
      application.onException := cgiErrorHandler;
      application.processMessages;

      {HTML page header info}
      createStdout;
      sendPrologue;

      send( '<HTML><HEAD>' );
      sendTitle( 'DecisionNet AMPL Execution' );
      send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
      sendHdr( '2','DecisionNet AMPL Solver Execution' );
      sendHdr('3','Choose one of the three methods below to use this
        solver.</CENTER>');
      sendHR;
    end;
  end;
end;

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin

```

```

with CGIEnvData1 do
begin
    {HTML page footer info}

    send( 'This application was created by Michael Casey for Professor
        ');
    send( 'Hemant Bhargava.<br>' );

    {timestamp}
    send( 'Generated on ' + webdate(now) );

    send( '</BODY></HTML>' );
    closeStdout;
    closeApp(application);
end;
end;

end.

```

AMPLEXE5

```
unit Ampexe5;

{ This program processes the user responses requested by AMPLEXE2. It
determines whether the user wishes to use a registered dataset or
provide his own. If an unregistered dataset is provided, it is stored
temporarily in the Tdataset table, and its DatasetNumber is prefixed
with a "t". This program also looks up the default solver for the chosen
model schema.
  INPUTS: Method, DatasetID, Dataset, ModelProvID, ModelTechID,
SolverURL
  OUTPUTS: ModelNumber, DatasetNumber, SolverURL
}

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgldb, Cgi, Grids, DBGrids, DB, DBTables, Ftp,
  starsock,
  StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    QuerySolver: TQuery;
    TblModel: TTable;
    TblDataSet: TTable;
    TblTDataSet: TTable;
    IniLink2: TIniFileLink;
    procedure SendHtmlHeader;
    procedure Form1Create(Sender: TObject);
    procedure SendHtmlFooter;
    procedure UseMethod1;
    procedure UseMethod2;
    procedure SendForm1;

  private
    { Private declarations }
  public
    { Public declarations }

  end;

var
  Form1: TForm1;

  ModelProvID : string;
  ModelTechID : string;
  Dataset      : TStringList;
  Method       : String;
  DSProvID     : String;
  DSTechID     : String;
  DatasetID    : String;
  ModelNumber, DataSetNumber, SolverURL : String;
  SolverProvID, SolverTechID : String;
```



```
implementation
{$R *.DFM}
```

```
procedure TForm1.Form1Create(Sender: TObject);
```

```
begin
```

```
  with CGIEnvData1 do
```

```
    begin
```

```
      {set up for CGI I/O}
```

```
      SendHtmlHeader;
```

```
      {retrieve input fields from HTML form}
```

```
      Method := getSmallField( 'Method' );
```

```
      DatasetID := getSmallField( 'DatasetID' );
```

```
      DSProVID := Copy(DatasetID, 0, Pos('/', DatasetID)-1 );
```

```
      DSTechID := Copy(DatasetID, Pos('/', DatasetID)+1, 10 );
```

```
      ModelProVID := getSmallField( 'ModelProVID' );
```

```
      ModelTechID := getSmallField( 'ModelTechID' );
```

```
      SolverURL := getSmallField( 'SolverURL' ); {present only if solver  
        chosen already}
```

```
      {get text for model, output script}
```

```
      Dataset := TStringList.create;
```

```
      Dataset.clear;
```

```
      GetTextArea( 'Dataset' , Dataset );
```

```
      {Get ModelNumber for chosen model}
```

```
      with TblModel do
```

```
        begin
```

```
          open;
```

```
          setKey;
```

```
          FieldByName('ProVID').AsString := ModelProVID;
```

```
          FieldByName('TechID').AsString := ModelTechID;
```

```
          gotoKey;
```

```
          ModelNumber := FieldByName('ModelNumber').AsString;
```

```
        end;
```

```
      {Use appropriate method to retrieve outputs}
```

```
      if Method = '1' then UseMethod1; {user chose a registered dataset}
```

```
      if Method = '2' then UseMethod2; {user chose to type/paste a
```

```
dataset}
```

```
      end;
```

```
end;
```

```
procedure TForm1.SendHtmlHeader;
```

```
{ This procedure sets up for CGI output}
```

```
begin
```

```
  with CGIEnvData1 do
```

```
    begin
```

```
      {required when this program runs under WebSite}
```

```
      webSiteINIFileName := paramstr(1);
```

```
      application.OnException := cgiErrorHandler;
```

```
      application.ProcessMessages;
```

```
      {HTML page header info}
```

```
      createStdout;
```

```

    sendPrologue;

    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Execution' );
    send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
    sendHdr( '2','DecisionNet AMPL Model Schema Execution' );
    send('</CENTER>');
    sendHR;
end;
end;

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin
    with CGIEnvData1 do
    begin
        {HTML page footer info}

        send( 'This application was created by Michael Casey for Professor
            ');
        send( 'Hemant Bhargava.<br>' );

        {timestamp}
        send( 'Generated on ' + webdate(now) );

        send( '</BODY></HTML>' );
        closeStdout;
        closeApp(application);
    end;
end;

procedure TForm1.UseMethod1;
begin
    with TblDataset do
    begin
        open;
        setKey;
        FieldByName('ProvID').AsString := DSProvID;
        FieldByName('TechID').AsString := DSTechID;
        gotoKey;
        DataSetNumber := FieldByName('DataSetNumber').AsString;
    end;

    if SolverURL = 'AAKEY NOT FOUND' then {user didn't already select a
        solver}
    {Get URL for model's default solver}
    begin
        SolverProvID := TblModel.FieldByName('SolverProvID').AsString;
        SolverTechID := TblModel.FieldByName('SolverTechID').AsString;
        if ((SolverProvID = '') and (SolverTechID = '')) then
        begin
            {Use system default solver}
            SolverProvID := IniLink2.StringEntry['DefaultSolverProvID'];
            SolverTechID := IniLink2.StringEntry['DefaultSolverTechID'];
        end;
        with QuerySolver do
        begin
            close;

```

```

        SQL.Clear;
        SQL.Add('SELECT URL FROM SOLVER WHERE');
        SQL.Add('(ProvID="' + SolverProvID + '"');
        SQL.Add('AND TechID="' + SolverTechID + '"')');
        open;
        SolverURL := FieldByName('URL').AsString;
    end;
end;

    SendForm1;
end;

procedure TForm1.UseMethod2;
begin
    {Put temporary dataset into TDATASET table}
    with TblTDataSet do
    begin
        open;
        AppendRecord(['', 'AMPL', Dataset, 'NO', '', DateTimeToStr(Now)]);
        DataSetNumber := 'T' + FieldByName('DataSetNumber').AsString;
    end;

    if SolverURL = 'AAKEY NOT FOUND' then {user didn't already select a
        solver}
    {Get URL for model's default solver}
    begin
        SolverProvID := TblModel.FieldByName('SolverProvID').AsString;
        SolverTechID := TblModel.FieldByName('SolverTechID').AsString;
        if ((SolverProvID = '') and (SolverTechID = '')) then
        begin
            {Use system default solver}
            SolverProvID := IniLink2.StringEntry['DefaultSolverProvID'];
            SolverTechID := IniLink2.StringEntry['DefaultSolverTechID'];
        end;
        with QuerySolver do
        begin
            close;
            SQL.Clear;
            SQL.Add('SELECT URL FROM SOLVER WHERE');
            SQL.Add('(ProvID="' + SolverProvID + '"');
            SQL.Add('AND TechID="' + SolverTechID + '"')');
            open;
            SolverURL := FieldByName('URL').AsString;
        end;
    end;
    SendForm1;
end;

procedure TForm1.SendForm1;
begin
    with CGIEnvData1 do
    begin
        send('<CENTER>');
        Send('<FORM method=post ' +
            'action="' + IniLink.StringEntry['VerificationExe'] + '>');
        send('<input type=hidden name="ModelNumber"
value="' + ModelNumber + '>');
        send('<input type=hidden name="DatasetNumber"
value="' + DataSetNumber + '>');
        send('<input type=hidden name="SolverURL" value="' + SolverURL + '>');
    end;
end;

```

```
    sendHdr('4','Your input has been accepted. Please press the button
to      continue.<BR>');
    send( '<INPUT TYPE=SUBMIT VALUE="OK" NAME="OK">' );
    send('</CENTER>');
    sendHR;

    SendHtmlFooter;
end;
end;

end.
```

AMPLEXE6

```
unit Ampexe6;
```

```
{ This program processes the user responses requested by AMPLEXE3. It
determines whether the user wishes to use a registered model schema or
provide his own. If an unregistered model schema is provided, it is
stored temporarily in the Tmodel table, and its ModelNumber is prefixed
with a "T". This program also looks up the default solver for the chosen
model schema.
```

```
  INPUTS: Method, ModelID, DefaultProvID, DefaultTechID, SolverID,
          SolverURL, model, output, DSProvID, DSTechID
```

```
  OUTPUTS: ModelNumber, DatasetNumber, SolverURL
```

```
}
```

```
interface
```

```
uses
```

```
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgldb, Cgi, Grids, DBGrids, DB, DBTables, Ftp,
  starsock,
```

```
  StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;
```

```
type
```

```
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanell1: TtpStatusPanel;
    IniLink: TIniFileLink;
    QuerySolver: TQuery;
    TblModel: TTable;
    TblDataSet: TTable;
    TblTModel: TTable;
    IniLink2: TIniFileLink;
    procedure SendHtmlHeader;
    procedure Form1Create(Sender: TObject);
    procedure SendHtmlFooter;
    procedure UseMethod1;
    procedure UseMethod2;
    procedure UseMethod3;
    procedure SendForm1;
    procedure SendNoDataMsg;
```

```
  private
```

```
    { Private declarations }
```

```
  public
```

```
    { Public declarations }
```

```
end;
```

```
var
```

```
  Form1: TForm1;
```

```
  ModelID      : string;
  Model, Output : TStringList;
  Method       : String;
  DSProvID     : String;
  DSTechID     : String;
  ModelNumber  : String;
```

```

    DataSetNumber : String;
    DefaultProvID : String;
    DefaultTechID : String;
    SolverID      : String;
    SolverURL     : String;
    SolverProvID, SolverTechID : String;

implementation

{$R *.DFM}

procedure TForm1.Form1Create(Sender: TObject);

begin
    with CGIEnvData1 do
        begin
            {set up for CGI I/O}
            SendHtmlHeader;

            {retrieve input fields from HTML form}
            Method := getSmallField( 'Method' );
            ModelID := getSmallField( 'ModelID' );
            DSProvID := getSmallField( 'DSProvID' );
            DSTechID := getSmallField( 'DSTechID' );
            ModelNumber := getSmallField( 'ModelNumber' );
            DefaultProvID := getSmallField( 'DefaultProvID' );
            DefaultTechID := getSmallField( 'DefaultTechID' );
            SolverURL := getSmallField( 'SolverURL' ); {present only if solver
                chosen already}

            {get text for model, output script}
            Model := TStringList.create;
            Model.clear;
            GetTextArea( 'model' , Model );

            Output := TStringList.create;
            Output.clear;
            GetTextArea( 'output' , Output );

            {Get DataSetNumber for chosen dataset}
            with TblDataset do
                begin
                    open;
                    setKey;
                    FieldByName('ProvID').AsString := DSProvID;
                    FieldByName('TechID').AsString := DSTechID;
                    gotoKey;
                    DataSetNumber := FieldByName('DataSetNumber').AsString;
                end;

            {Use appropriate method to retrieve outputs}
            if Method = '1' then UseMethod1; {user chose the default model
schema}
            if Method = '2' then UseMethod2; {user chose another registered
model          schema}
            if Method = '3' then UseMethod3; {user chose to type/paste a model
schema}
            end;
        end;
    end;
end;

```

```

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin
  with CGIEnvData1 do
  begin
    {required when this program runs under WebSite}
    webSiteINIFileName := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {HTML page header info}
    createStdout;
    sendPrologue;

    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Execution' );
    send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
    sendHdr( '2','DecisionNet AMPL Dataset Execution' );
    send('</CENTER>');
    sendHR;
  end;
end;

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin
  with CGIEnvData1 do
  begin
    {HTML page footer info}

    send( 'This application was created by Michael Casey for Professor
    ');
    send( 'Hemant Bhargava.<br>' );

    {timestamp}
    send( 'Generated on ' + webdate(now) );

    send( '</BODY></HTML>' );
    closeStdout;
    closeApp(application);
  end;
end;

procedure TForm1.UseMethod1; {user chose the default model schema}
begin
  {Get ModelNumber for default model schema}
  with TblModel do
  begin
    open;
    setKey;
    FieldByName('ProvID').AsString := DefaultProvID;
    FieldByName('TechID').AsString := DefaultTechID;
    gotoKey;
    ModelNumber := FieldByName('ModelNumber').AsString;
  end;

  if SolverURL = 'AAAKEY NOT FOUND' then {user didn't already select a
  solver}
  {Get URL for model's default solver}

```

```

begin
  SolverProvID := TblModel.FieldName('SolverProvID').AsString;
  SolverTechID := TblModel.FieldName('SolverTechID').AsString;
  if ((SolverProvID = '') and (SolverTechID = '')) then
  begin
    {Use system default solver}
    SolverProvID := IniLink2.StringEntry['DefaultSolverProvID'];
    SolverTechID := IniLink2.StringEntry['DefaultSolverTechID'];
  end;
  with QuerySolver do
  begin
    close;
    SQL.Clear;
    SQL.Add('SELECT URL FROM SOLVER WHERE');
    SQL.Add('(ProvID="' + SolverProvID + '"');
    SQL.Add('AND TechID="' + SolverTechID + '"');
    open;
    SolverURL := FieldByName('URL').AsString;
  end;
end;
SendForm1;
end;

procedure TForm1.UseMethod2; {user chose another registered model
schema}
begin
  {Get ModelNumber for selected model schema}
  with TblModel do
  begin
    open;
    setKey;
    FieldByName('ProvID').AsString := Copy(ModelID, 0, Pos('/',
ModelID)-1
    );
    FieldByName('TechID').AsString := Copy(ModelID, Pos('/', ModelID)+1
    ,
    10 );
    gotoKey;
    ModelNumber := FieldByName('ModelNumber').AsString;
  end;

  if SolverURL = 'AAKEY NOT FOUND' then {user didn't already select a
solver}
  {Get URL for model's default solver}
  begin
    SolverProvID := TblModel.FieldName('SolverProvID').AsString;
    SolverTechID := TblModel.FieldName('SolverTechID').AsString;
    if ((SolverProvID = '') and (SolverTechID = '')) then
    begin
      {Use system default solver}
      SolverProvID := IniLink2.StringEntry['DefaultSolverProvID'];
      SolverTechID := IniLink2.StringEntry['DefaultSolverTechID'];
    end;
    with QuerySolver do
    begin
      close;
      SQL.Clear;
      SQL.Add('SELECT URL FROM SOLVER WHERE');
      SQL.Add('(ProvID="' + SolverProvID + '"');
      SQL.Add('AND TechID="' + SolverTechID + '"');
      open;
      SolverURL := FieldByName('URL').AsString;

```



```

        end;
    end;
    SendForm1;
end;

procedure TForm1.UseMethod3; {user chose to type/paste a model schema}
begin
    {Put temporary model into TMODEL table}
    with TblTModel do
    begin
        open;
        AppendRecord(['','AMPL', Model, Output, 'NO', DateTimeToStr(Now)]);
        ModelNumber := 'T' + FieldByName('ModelNumber').AsString;
        close;
    end;

    if SolverURL = 'AAKEY NOT FOUND' then {user didn't start by selecting
a    solver}
        with QuerySolver do
        {Get URL for solver user chose while specifying temporary model
schema}
        begin
            close;
            SQL.Clear;
            SQL.Add('SELECT URL FROM SOLVER WHERE');
            SQL.Add('(ProvID="' + Copy(SolverID, 0, Pos('/', SolverID)-1 ) +
            '"');
            SQL.Add('AND TechID="' + Copy(SolverID, Pos('/', SolverID)+1 , 10 )
+            '"')');
            open;
            SolverURL := FieldByName('URL').AsString;
        end;
        SendForm1;
    end;

procedure TForm1.SendForm1;
begin
    with CGIEnvData1 do
    begin
        send('<CENTER>');
        Send('<FORM method=post ' +
        'action="' + IniLink.StringEntry['VerificationExe'] + '">');
        send('<input type=hidden name="ModelNumber"
value="'+ModelNumber+'">');
        send('<input type=hidden name="DatasetNumber"
value="'+DataSetNumber+'">');
        send('<input type=hidden name="SolverURL" value="'+SolverURL+'">');
        sendHdr('4','Your input has been accepted. Please press the button
to    continue.<BR>');
        send( '<INPUT TYPE=SUBMIT VALUE="OK" NAME="OK">' );
        send('</CENTER>');
        sendHR;

        SendHtmlFooter;
    end;
end;

procedure TForm1.SendNoDataMsg;
begin

```

```

{Informs user that no metainformation is available}
with CGIEnvData1 do
begin
  send('<CENTER>');
  sendHdr('3','This option is not available');
  send('The provider of the model schema you chose did not provide ');
  send('the necessary information on its data structure.<BR>');
  send('You may use your browser BACK button to return to the previous
    form, '+
      ' from which you can view the model schema ');
  send('and type in a your own dataset for it. ');
  send('<BR><BR>');
  send('</CENTER>');
end;
  SendHtmlFooter;
end;
end.

```

AMPLEXE7

unit Ampexe7;

{ This program processes the user responses requested by AMPLEXE4. It determines whether the user wishes to use a registered model schema or dataset, or provide his own. If an unregistered model schema and dataset are provided, they are stored temporarily in the TModel and TDataset tables and AMPLEXE9 is called. If a registered model schema is chosen, it calls AMPLEXE2. If a registered dataset is chosen, it calls AMPLEXE3.

INPUTS: Method, SolverURL, ModelID, DatasetID, model, dataset, output
OUTPUTS: ModelNumber, DatasetNumber, SolverURL, ProviderID, TechID

}

interface

uses

SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
Forms, Dialogs, Cgldb, Cgi, Grids, DBGrids, DB, DBTables, Ftp,
starsock,
StdCtrls, Toolbar, ExtCtrls, UpdateOk, IniLink;

type

TForm1 = class(TForm)
CGIEnvData1: TCGIEnvData;
TpStatusBar1: TtpStatusBar;
TpStatusPanel1: TtpStatusPanel;
IniLink: TIniFileLink;
TblTModel: TTable;
TblTDataset: TTable;
procedure SendHtmlHeader;
procedure Form1Create(Sender: TObject);
procedure SendHtmlFooter;
procedure UseMethod1;
procedure UseMethod2;
procedure UseMethod3;

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

Model, Dataset, Output : TStringList;
Method : String;
ModelID : String;
DatasetID : String;
SolverURL : String;
ModelNumber, DatasetNumber : String;

implementation

{ \$R *.DFM }

procedure TForm1.Form1Create(Sender: TObject);

```

begin
  with CGIEnvData1 do
  begin
    {set up for CGI I/O}
    SendHtmlHeader;

    {retrieve input fields from HTML form}
    Method := getSmallField( 'Method' );
    DatasetID := getSmallField( 'DatasetID' );
    ModelID := getSmallField( 'ModelID' );
    SolverURL := getSmallField( 'SolverURL' );

    {get text for model, dataset, output script}
    Model := TStringList.create;
    Model.clear;
    GetTextArea( 'model' , Model );

    Dataset := TStringList.create;
    Dataset.clear;
    GetTextArea( 'dataset' , Dataset );

    Output := TStringList.create;
    Output.clear;
    GetTextArea( 'output' , Output );

    {Use appropriate method to retrieve outputs}
    if Method = '1' then UseMethod1; {user chose a registered model}
    if Method = '2' then UseMethod2; {user chose a registered dataset}
    if Method = '3' then UseMethod3; {user chose to supply model &
dataset}
  end;
end;

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin
  with CGIEnvData1 do
  begin
    {required when this program runs under WebSite}
    webSiteINIFileName := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {HTML page header info}
    createStdout;
    sendPrologue;

    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Execution' );
    send( '</HEAD><BODY BGCOLOR="80B7B0"><CENTER>' );
    sendHdr( '2','DecisionNet AMPL Solver Execution' );
    send('</CENTER>');
    sendHR;
  end;
end;

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin

```

```

with CGIEnvData1 do
begin
    {HTML page footer info}

    send( 'This application was created by Michael Casey for Professor
    ');
    send( 'Hemant Bhargava.<br>' );

    {timestamp}
    send( 'Generated on ' + webdate(now) );

    send( '</BODY></HTML>' );
    closeStdout;
    closeApp(application);
end;
end;

procedure TForm1.UseMethod1;
begin
    with CGIEnvData1 do
    begin
        send('<CENTER>');
        Send('<FORM method=post ' +
        'action="' + IniLink.StringEntry['ModelExe'] + '">');
        send('<input type=hidden name="SolverURL" value="'+SolverURL+'">');
        send('<input type=hidden name="ProviderID" value="' +
        Copy(ModelID, 0, Pos('/', ModelID)-1 ) + '">');
        send('<input type=hidden name="TechID" value="' +
        Copy(ModelID, Pos('/', ModelID)+1 , 10 ) + '">');
        sendHdr('4','Your selection has been accepted. Please press the
        button to continue.<BR>');
        send( '<INPUT TYPE=SUBMIT VALUE="OK" NAME="OK">' );
        send('</CENTER>');
        sendHR;

        SendHtmlFooter;
    end;
end;

procedure TForm1.UseMethod2;
begin
    with CGIEnvData1 do
    begin
        send('<CENTER>');
        Send('<FORM method=post ' +
        'action="' + IniLink.StringEntry['DatasetExe'] + '">');
        send('<input type=hidden name="SolverURL" value="'+SolverURL+'">');
        send('<input type=hidden name="ProviderID" value="' +
        Copy(DatasetID, 0, Pos('/', DatasetID)-1 ) + '">');
        send('<input type=hidden name="TechID" value="' +
        Copy(DatasetID, Pos('/', DatasetID)+1 , 10 ) + '">');
        sendHdr('4','Your selection has been accepted. Please press the
        button to continue.<BR>');
        send( '<INPUT TYPE=SUBMIT VALUE="OK" NAME="OK">' );
        send('</CENTER>');
        sendHR;

        SendHtmlFooter;
    end;
end;
end;

```

```

procedure TForm1.UseMethod3;
begin
  with CGIEnvData1 do
  begin
    send('<CENTER>');
    Send('<FORM method=post ' +
      'action="' + IniLink.StringEntry['VerificationExe'] + '">');
    {Put temporary model into TMODEL table}
    with TblTModel do
    begin
      open;
      AppendRecord(['','AMPL', Model, Output, 'NO',
DateTImeToStr(Now)]);
      ModelNumber := 'T' + FieldByName('ModelNumber').AsString;
      close;
    end;

    {Put temporary dataset into TDATASET table}
    with TblTDataset do
    begin
      open;
      AppendRecord(['','AMPL', Dataset, 'NO', DateTImeToStr(Now)]);
      DatasetNumber := 'T' + FieldByName('DataSetNumber').AsString;
      close;
    end;

    send('<input type=hidden name="SolverURL" value="'+SolverURL+'">');
    send('<input type=hidden name="ModelNumber" value="' +
      ModelNumber + '">');
    send('<input type=hidden name="DatasetNumber" value="' +
      DatasetNumber + '">');
    sendHdr('4','Your input has been accepted. Please press the button
to      continue.<BR>');
    send( '<INPUT TYPE=SUBMIT VALUE="OK" NAME="OK">' );
    send('</CENTER>');
    sendHR;

    SendHtmlFooter;
  end;
end;

end.

```

AMPLEXE9

```
unit Ampexe9;
```

```
{ This program allows the user to verify the model schema, dataset, and
  execution script just before they are sent to the AMPL solver. It
  retrieves
  the execution script from either the Model or TModel table. When the
  user
  submits the form generated by this program, the AMPL solver residing
  at the
  SolverURL is called.
  INPUTS: ModelNumber, DatasetNumber, SolverURL
  OUTPUTS: model, dataset, output, FileStub
}
```

```
interface
```

```
uses
```

```
SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
Forms, Dialogs, DB, DBTables, Cgi, Cgidb, UpdateOk, IniLink, StdCtrls;
```

```
type
```

```
TForm1 = class(TForm)
  CGIEnvData1: TCGIEnvData;
  ModelQuery: TQuery;
  DataSetQuery: TQuery;
  SolverQuery: TQuery;
  CGIDB1: TCGIDB;
  IniLink: TIniFileLink;

  procedure SendHtmlHeader;
  procedure Form1Create(Sender: TObject);
  procedure SendHtmlFooter;
  procedure VerifyFiles;
  procedure EraseOldFiles;
  function GetTempFileName: string;
```

```
private
  { Private declarations }
public
  { Public declarations }
end;
```

```
var
```

```
Form1: TForm1;

TempFile : TextFile;
ModelNumber : String;
DataSetNumber : String;
SolverURL : String;
```

```
implementation
```

```
{ $R *.DFM }
```

```
procedure TForm1.Form1Create(Sender: TObject);
```

```
begin
```

```
  with CGIEnvData1 do
```

```

begin
  {set up for CGI I/O}
  SendHtmlHeader;

  {retrieve input fields from HTML form}
  ModelNumber := getSmallField( 'ModelNumber' );
  DataSetNumber := getSmallField( 'DataSetNumber' );
  SolverURL := getSmallField( 'SolverURL' );

  {Setup query to retrieve model}
  ModelQuery.Close;
  ModelQuery.SQL.Clear;
  if Copy(ModelNumber,0,1) = 'T' then {model is in TMODEL table}
    ModelQuery.SQL.Add( 'SELECT * FROM TMODEL WHERE ModelNumber = ' +
      Copy(ModelNumber,2,9) )
  else
    ModelQuery.SQL.Add( 'SELECT * FROM MODEL WHERE ModelNumber = ' +
      ModelNumber );
  ModelQuery.Open;

  {Setup query to retrieve dataset}
  DataSetQuery.Close;
  DataSetQuery.SQL.Clear;
  if Copy(DataSetNumber,0,1) = 'T' then {model is in TMODEL table}
    DataSetQuery.SQL.Add( 'SELECT * FROM TDATASET WHERE DataSetNumber
=      ' +
      Copy(DataSetNumber,2,9) + ';' )
  else
    DataSetQuery.SQL.Add( 'SELECT * FROM DATASET WHERE DataSetNumber =
      ' +
      DataSetNumber + ';' );
  DataSetQuery.Open;

  {send HTML form to verify model, dataset, output script}
  VerifyFiles;
end;
end;

procedure TForm1.SendHtmlHeader;
{ This procedure sets up for CGI output}
begin
  with CGIEnvData1 do
  begin
    {required when this program runs under WebSite}
    webSiteINIFileName := paramstr(1);
    application.onException := cgiErrorHandler;
    application.processMessages;

    {HTML page header info}
    createStdout;
    sendPrologue;

    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Execution' );
    send( '</HEAD><BODY BGCOLOR="80B7B0">' );
    sendHdr( '2','DecisionNet AMPL Execution' );
    sendHR;
  end;
end;

```



```

procedure TForm1.SendHtmlFooter;
{ This procedure sends the HTML footer and closes the app. }
begin
  with CGIEnvData1 do
  begin
    {HTML page footer info}
    send('<HR>');
    send( 'This application was created by Michael Casey for Professor
      ');
    send( 'Hemant Bhargava.<br>' );

    {timestamp}
    send( 'Generated on ' + webdate(now) );

    send( '</BODY></HTML>' );
    closeStdout;
    closeApp(application);
  end;
end;

```

```

procedure TForm1.VerifyFiles;
{Allows user to verify/modify all 3 files. When form is submitted, AMPL
 solver agent is called}
begin
  with CGIEnvData1 do
  begin
    {HTML page header info}
    createStdout;
    sendPrologue;
    send( '<HTML><HEAD>' );
    sendTitle( 'DecisionNet AMPL Execution' );
    send( '</HEAD><BODY BGCOLOR="80B7B0">' );
    send( '<CENTER><h1>DecisionNet AMPL Execution</h1></CENTER>' );
    sendHR;

    {setup HTML form (which calls solver agent when submitted) }
    send( '<FORM method=post' +
      ' action="' + SolverURL + '">' );
    send( '<CENTER>' );

    {send form memo field containing model}
    SendHdr('3','Please verify the model schema, correcting it if
necessary.' );
    send( '<textarea name="model" rows=20 cols=80>' );
    CGIDB1.sendMemo( ModelQuery.FieldName('Model') );
    send( '</textarea><BR><BR><HR>' );

    {send form memo field containing dataset}
    SendHdr('3','Please verify the dataset, correcting it if necessary.'
    );
    send( '<textarea name="dataset" rows=20 cols=80>' );
    CGIDB1.sendMemo( DataSetQuery.FieldName('Dataset') );
    send( '</textarea><BR><BR><HR>' );

    {send form memo field containing script}
    SendHdr('3','Please verify the execution script, correcting it if
necessary.' );

```

```

send( '<textarea name="output" rows=20 cols=80>' );
CGIDB1.sendMemo( ModelQuery.FieldByName('Output') );
send( '</textarea><BR><BR>' );

{send hidden field containing output filename stub}
EraseOldFiles; {Gets rid of all output files over 1 hr old}
send( '<INPUT TYPE=HIDDEN NAME="FileStub" VALUE="' +
      GetTempFileName + '>' );

{Finish HTML form}
send( '<INPUT TYPE=SUBMIT VALUE="Submit">' );
send( '<INPUT TYPE=RESET VALUE="Reset Form">' );
send( '</CENTER></FORM>' );

SendHtmlFooter;
end;
end;

function TForm1.GetTempFileName: string;
{Finds filename stub for next output file}
var
  StubNumber : LongInt;
begin
  StubNumber := StrToInt(IniLink.StringEntry['LastStubNo']);
  If StubNumber = 99999 then
    StubNumber := 1
  else
    StubNumber := StubNumber + 1;
  IniLink.StringEntry['LastStubNo'] := IntToStr( StubNumber );
  GetTempFileName := 'aof' + IntToStr( StubNumber );
end;

procedure TForm1.EraseOldFiles;
{Erases all stored AMPL output files over one hour old}
var
  SearchRec: TSearchRec;
  OneHour : tDateTime;
begin
  OneHour := 1.0/24.0 ; {Equates to one hour in tDateTime format}
  If FindFirst(IniLink.StringEntry['output_path']+'aof*.txt', faAnyFile,
SearchRec) = 0 then
    {at least one output file exists}
    begin
      If (Now-FileDateToDateTime(SearchRec.Time)) > OneHour then {File has
expired}
        DeleteFile(IniLink.StringEntry['output_path'] + SearchRec.Name);
        while FindNext(SearchRec) = 0 do {for each subsequent file}
          If (Now-FileDateToDateTime(SearchRec.Time)) > OneHour then {File
has expired}
            DeleteFile(IniLink.StringEntry['output_path'] + SearchRec.Name);
        end;
      end;
    end;
end.

```

AMPLCALL

```

unit Amplcall;

interface

uses
  SysUtils, WinTypes, WinProcs, Messages, Classes, Graphics, Controls,
  Forms, Dialogs, Cgi, StdCtrls, Toolbar, ExtCtrls,
  UpdateOk, TpShell, IniLink, Ftp, starsock;

type
  TForm1 = class(TForm)
    CGIEnvData1: TCGIEnvData;
    WindowsShell1: TWindowsShell;
    TpStatusBar1: TtpStatusBar;
    TpStatusPanel1: TtpStatusPanel;
    IniLink: TIniFileLink;
    StarSocket1: TStarSocket;
    Ftp1: TFtp;
    procedure ProcessHtmlForm(Sender: TObject);
    procedure SendHtmlHeader;
    procedure SendHtmlFooter;
    procedure DoFtpUpload;
    procedure SendErrorMsg(ErrorDesc : String);
    function GetTempFileName: string;
    procedure SendHtmlOutput(Sender: TObject);
    procedure Ftp1FtpError(Sender: TObject; error: FtpError;
      addinfo: String);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  Form1: TForm1;
  ModelName : string;
  DatasetName : string;
  BatchName : string;
  TempFileName : String;
  TempFilePath : String;
  TempOutputPath : String;
  OutputFileStub : String;

implementation

{$R *.DFM}

procedure TForm1.ProcessHtmlForm(Sender: TObject);
var
  ModelStringList : TStringList;
  DataSetStringList : TStringList;
  OutputStringList : TStringList;
  BatchFile : TextFile;
begin
  with CGIEnvData1 do

```

```

begin
  {required when this program runs under WebSite}
  webSiteINIFileName := paramstr(1);
  application.OnException := cgiErrorHandler;
  application.ProcessMessages;

  {send header for cgi output to client}
  SendHtmlHeader;

  {find temporary file path and next unused temporary filename}
  TempFilePath := IniLink.StringEntry['tempfilepath'];
  TempOutputPath := IniLink.StringEntry['tempoutputpath'];
  TempFileName := GetTempFileName;

  {get text for model, dataset, output script}
  ModelStringList := TStringList.Create;
  ModelStringList.Clear;
  if GetTextArea( 'model' , ModelStringList ) then
    ModelStringList.SaveToFile( TempFilePath + TempFileName + '.MOD'
  )
  else
    SendErrorMsg( 'Unable to process model.' );

  DataSetStringList := TStringList.Create;
  DataSetStringList.Clear;
  if GetTextArea( 'dataset' , DataSetStringList ) then
    DataSetStringList.SaveToFile( TempFilePath + TempFileName +
'.DAT'
    )
  else
    SendErrorMsg( 'Unable to process dataset.' );

  OutputStringList := TStringList.Create;
  OutputStringList.Clear;
  if GetTextArea( 'output' , OutputStringList ) then
    begin
      OutputStringList.Insert(0, 'model ' + TempFilePath + TempFileName
        + '.mod;');
      OutputStringList.Insert(1, 'data ' + TempFilePath + TempFileName
        + '.dat;');
      OutputStringList.SaveToFile( TempFilePath + TempFileName + '.RUN'
        );
    end
  else
    SendErrorMsg( 'Unable to process output script.' );

  {Get filename stub for ftp'ing output file to AMPL agent}
  OutputFileStub := getSmallField( 'FileStub' );

  {write a batch file for the AMPL executable's command line}
  AssignFile( BatchFile, TempFilePath + TempFileName + '.bat' );
  Rewrite( BatchFile );
  WriteLn( BatchFile, '@echo off' );
  WriteLn( BatchFile, 'echo "<HTML><PRE>"***** OUTPUT FOLLOWS ***** >
+
    TempFilePath + TempFileName + '.htm' );
+
  WriteLn( BatchFile, IniLink.StringEntry['ampl_exe_filename'] + ' '
    TempFilePath + TempFileName + '.run' + ' >> ' + TempFilePath +
    TempFileName + '.htm');

```

```

    WriteLn( BatchFile, 'echo "</PRE></HTML>" >> ' +
      TempFilePath + TempFileName + '.htm' );
    CloseFile( BatchFile );

    {call AMPL solver and send solution to temporary output file}
    WindowsShell1.Command := TempFilePath + TempFileName + '.bat';
    try
      WindowsShell1.Execute;
    except
      SendErrorMsg( 'Unable to execute the AMPL solver.' );
    end;

    {Send output file to Dnet AMPL Agent via ftp}
    DoFtpUpload;
  end;
end;

procedure TForm1.SendHtmlOutput(Sender: TObject);
begin
  with CGIEnvData1 do
    begin
      {send HTML message with link to output file}
      sendHdr('3','The AMPL solver is finished');
      send( '<h4>The ' );
      sendHREF( '', '', 'output file', IniLink.StringEntry['output_url'] +
        OutputFileStub + '.htm' );
      send( ' is available for download.<br>' );
      send( 'Note: This output file will be deleted from the server in ' +
        'approximately one hour.' );

      {Delete temp files, except output file}
      DeleteFile( TempFilePath + TempFileName + '.bat' );
      DeleteFile( TempFilePath + TempFileName + '.mod' );
      DeleteFile( TempFilePath + TempFileName + '.dat' );
      DeleteFile( TempFilePath + TempFileName + '.run' );
      DeleteFile( TempFilePath + TempFileName + '.htm' );

      SendHtmlFooter;
    end;
  end;

procedure TForm1.SendErrorMsg(ErrorDesc : String);
{sends HTML error message}
begin
  with CGIEnvData1 do
    begin
      sendHdr( '3','The AMPL Solver Agent was unable to provide the
        requested solution. ');
      sendHdr( '3','Reason: ' + ErrorDesc );
      sendHdr( '4','Use your browser "Back" button to try again.' );

      SendHtmlFooter;
    end;
  end;

function TForm1.GetTempFileName: string;
{Finds first unused filename (extensionless) for temp files}
var
  Index : Integer;

```

```

begin
  Index := 1;
  while FileExists( TempOutputPath + 'ampl' + IntToStr(Index) + '.htm' )
  do
    Index := Index + 1;
    GetTempFileName := 'ampl' + IntToStr(Index);
  end;

  procedure TForm1.SendHtmlHeader;
  begin
    with CGIEnvData1 do
    begin
      {HTML page header info}
      createStdout;
      sendPrologue;

      send( '<HTML><HEAD>' );
      sendTitle( 'DecisionNet AMPL Solver' );
      send( '</HEAD><BODY BGCOLOR="80B7B0">' );
      sendHdr( '2','DecisionNet AMPL Solver' );
      sendHR;

    end;
  end;

  procedure TForm1.SendHtmlFooter;
  begin
    with CGIEnvData1 do
    begin
      {HTML page footer info}
      send('<HR>');
      send( 'This application was created by Michael Casey for Professor
        ' );
      send( 'Hemant Bhargava.<br>' );
      send( 'Generated on ' + webdate(now) );
      send( '</BODY></HTML>' );
      closeStdout;

      {quit application}
      closeApp(application);
    end;
  end;

  procedure TForm1.DoFtpUpload;
  begin
    {Sends output text file via ftp}
    with ftp1 do
    begin
      UserName := IniLink.StringEntry['userid'];
      UserPassword := IniLink.StringEntry['password'];
      Ftp_URL := IniLink.StringEntry['ftp_url'] + OutputFileStub + '.htm';
      LocalFile := TempFilePath + TempFileName + '.htm';
      PutURL;
    end;
  end;
end;

```

```
procedure TForm1.Ftp1FtpError(Sender: TObject; error: FtpError;  
    addinfo: String);  
begin  
    SendErrorMsg('An FTP error occurred while attempting to upload  
results.');
```

end;

end.

LIST OF REFERENCES

1. Earley, S. H., *DecisionNet: A Database Approach*, Master's Thesis, Naval Postgraduate School, Monterey, California, September 1996.
2. Bhargava, H. K., Krishnan, R., and Muller, R., "On Parameterized Transaction Models for Agents in Electronic Markets for Decision Technologies," *Proceedings of the Fifth WITS Amsterdam, Holland*, 1995.
3. Fourer, R., Gay, D. M., and Kernighan, B. W., *AMPL: A Modeling Language for Mathematical Programming*, The Scientific Press, 1993.
4. Whitten, L. W., Bentley, L. D., and Barlow, V. M., *Systems Analysis and Design Methods*, Richard D. Irwin, Inc, 1994.
5. Bhargava, H. K., and others, "Model Management in Electronic Markets for Decision Technologies: A Software Agent Approach," *Proceedings of the Thirteenth Annual Hawaii International Conference on System Sciences*, 1997.
6. Kroenke, D. M., *Database Processing*, Prentice Hall, 1995.
7. Brooke, A., Kendrick, D., and Meeraus, A., *GAMS: A User's Guide, Release 2.25*, The Scientific Press, 1993.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center2
 8725 John J. Kingman Rd, STE 0944
 Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library2
 Naval Postgraduate School
 411 Dyer Rd.
 Monterey, CA 93943-5101

3. Prof. Hemant Bhargava, Code SM/Bh2

4. Prof. Gordon Bradley, Code OR/Bz1

5. Prof. Arthur Geoffrion.....1
 Decision Sciences
 John E. Anderson Graduate School of Management
 Box 951481, UCLA
 Los Angeles, CA 90095-1481

6. Lt. Michael Casey, USN1
 USS TREPANG (SSN 674)
 FPO AP 96679-2354